ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO EN INFORMÁTICA

Título del proyecto:

GENERALIZACIÓN DEL ALGORITMO DE CLÚSTER GRAVITACIONAL UTILIZANDO FUNCIONES DE OVERLAP

Iosu Rodríguez Martínez

Francisco Javier Fernández Fernández

Pamplona, 18/06/2018

ABSTRACT

El interés en las técnicas de clúster ha sufrido un crecimiento notable a lo largo de los últimos años, especialmente motivado por la necesidad de adquirir conocimiento a partir de enormes cantidades de datos para los cuales, a menudo se conocen poco más que una serie de valores carentes de contexto.

Multitud de algoritmos han sido propuestos para resolver este problema. En este caso, nos centraremos en una modificación del algoritmo de clúster gravitacional propuesto por Wright, que utiliza el concepto de función de overlap para mejorar su eficacia. Se analizarán los cambios introducidos y se compararán los resultados con algoritmos tan extendidos como son el K-means o el FCM, con el objetivo de descubrir los puntos fuertes y las carencias de la nueva versión.

Palabras clave: Clúster, overlap, fuerza gravitacional

1.-INTRODUCCIÓN 4
1.1.-Diferencia entre aprendizaje supervisado y no supervisado 4
1.2.-Objetivo: Generalización del algoritmo gravitacional para clúster 5
1.3.-Uso de overlaps 5
2.-ESTADO DEL ARTE 6
2.1.-Tipos de algoritmos 6
2.2.-Algoritmo K-means 9
2.3.-Algoritmo FCM (Fuzzy c-Means) 9
2.4.-Medidas de evaluación 11
3.-PRELIMINARES 14
3.1.-Funciones de agregación 14
3.2.-Funciones de overlap 14
3.3.- El algoritmo de clúster gravitacional 16
4.-UN NUEVO ALGORITMO DE CLÚSTER GRAVITACIONAL 18
5.-EXPERIMENTACIÓN 20
5.1.-Aproximación inicial: Overlaps sobre masas normalizadas 20
5.2.-Pruebas con overlaps sobre masas sin normalizar 22
5.3 Comportamiento del nuevo algoritmo 44
6.- CONCLUSIONES 49
7.- ANEXO: Propuesta de artículo derivado a partir de los resultados 52

1.-INTRODUCCIÓN

El interés en las técnicas de clúster ha sufrido un crecimiento notable a lo largo de los últimos años, especialmente motivado por la necesidad de adquirir conocimiento a partir de enormes cantidades de datos para los cuales, a menudo se conocen poco más que una serie de valores carentes de contexto. Desde sistemas de segmentación de clientes hasta reconocimiento de patrones en imágenes [1], pasando por minería de textos [2], planificación urbanística [3] o clasificación de genotipos [4], los algoritmos de clúster juegan un papel decisivo a la hora de encontrar soluciones efectivas a un gran número de problemas de muy diversa índole.

Debido a la amplia variedad de formas que pueden adoptar los datos a tratar para cada caso concreto, se han propuesto decenas de algoritmos distintos basados en otras tantas ideas. En concreto, en este caso nos centraremos en un algoritmo basado en la ley de gravitación universal y en las distintas modificaciones que pueden aplicársele con el fin de mejorar su rendimiento.

1.1.-Diferencia entre aprendizaje supervisado y no supervisado

Cuando se busca extraer conocimiento a partir de una serie de datos, existen dos acercamientos fundamentales, similares en apariencia aunque muy distintos en el modo de funcionamiento y la serie de problemas que cada uno permite abordar.

Por un lado están los métodos basados en aprendizaje supervisado. Se conoce con este nombre a aquellos algoritmos que trabajan con una serie de ejemplos para los que se cuenta con cierta información externa a los mismos, que resulta imprescindible para su correcto funcionamiento. Dicha información puede darse en forma de “etiquetas”, como sucede en los problemas de clasificación, o como valores numéricos, como es el caso de los problemas de regresión. Los algoritmos que forman parte de este conjunto “aprenden” de los datos del problema en el sentido de que buscan devolver, para cada ejemplo, la etiqueta o valor que se le ha asociado a priori. Suelen ser algoritmos iterativos que se “entrenan” constantemente con los mismos ejemplos, buscando minimizar el número de errores entre la salida esperada y la devuelta por el algoritmo, con el objetivo final de que este funcione correctamente al tratar con ejemplos con los que aún no ha trabajado.

Por el otro tenemos el aprendizaje no supervisado. En este caso se carece de la información externa con la que trabajan los algoritmos de aprendizaje supervisado, y se cuenta únicamente con los ejemplos del problema. Es por esto que la finalidad de los algoritmos suele ser encontrar ciertas similitudes entre los datos, generando grupos entre aquellos que se encuentren más estrechamente relacionados. Al no conocer las clasificaciones que se espera para el problema, surge también la necesidad de definir otro tipo de medidas de evaluación que indiquen si la división realizada tiene sentido. Por lo general, el criterio a seguir es buscar crear clústeres a partir de ejemplos parecidos, lo más diferentes posibles de los ejemplos del resto de clústeres.

A raíz de lo anterior, es fácil ver cómo los problemas a resolver con cada tipo de método son muy distintos. Mientras que en el aprendizaje supervisado se suele trabajar buscando objetivos muy concretos definidos en el entorno del problema, como puede ser predecir las ventas futuras de un tipo de producto en base a un histórico previo, el objetivo en el no supervisado suele ser más general, por ejemplo agrupar los clientes en grupos en función

de sus gustos, o comprobar si dicha agrupación es siquiera posible. Los métodos no supervisados funcionan a menudo como paso previo a los supervisados.

1.2.-Objetivo: Generalización del algoritmo gravitacional para clúster

El objetivo de este trabajo es recuperar el algoritmo basado en la ley de gravitación universal de Newton, propuesto por W. E. Wright [5], y mejorar los resultados obtenidos estudiando el efecto que tienen ciertos cambios sobre el mismo.

Con este fin en mente se probarán distintas variaciones del algoritmo, junto a algunos de los algoritmos más utilizados a día de hoy, en distintos datasets (tanto reales como sintéticos). Se comparará los resultados en base a ciertas medidas de evaluación y se analizará el comportamiento del algoritmo gravitacional teniendo en cuenta características como la topología de los datasets.

Tanto el algoritmo original como los cambios introducidos se presentarán más adelante.

1.3.-Uso de overlaps

En concreto, el primer cambio propuesto para el algoritmo es el de generalizar una de sus fórmulas fundamentales, sustituyendo un término concreto por una función de overlap, como se verá después.

Las funciones de overlap han demostrado ofrecer buenos resultados en otros campos, como son el del procesamiento de imagen o el de los problemas de clasificación, por lo que probar su efectividad sobre un algoritmo de clúster puede resultar de interés. Además, en nuestro caso la expresión a sustituir puede verse, de hecho, como el overlap mínimo o producto, por lo que su generalización parece tener sentido desde un punto de vista teórico. Esto significa que, en cualquier caso, el algoritmo clásico actúa como un caso particular de la generalización a presentar.

2.-ESTADO DEL ARTE

Tal y como ya se ha comentado, existen decenas de algoritmos que tratan de abordar un problema de clúster mediante distintos acercamientos y técnicas. Tal diversidad tiene su fundamento en el teorema "no free lunch" [6], que indica que un algoritmo que obtenga buenos resultados sobre un conjunto de problemas habrá de pagar su rendimiento en otro número igual de problemas. En este caso, el correcto funcionamiento de las distintas estrategias de clúster viene condicionado por los datasets que aborden, las exigencias temporales de cada uno o la cantidad de elementos con los que pueden tratar de manera eficiente.

Así, por ejemplo, mientras que en un dataset compuesto por series de elementos que formen cadenas separadas, un algoritmo basado en "single-linkage" obtendrá resultados mucho mejores que los devueltos por el algoritmo K-means, en otro dataset compuesto por clústeres esféricos la validez de los resultados se verá invertida [7].

A continuación se muestra un breve resumen de las principales estrategias de clústeres utilizadas [8] [9] [10], y se presenta en mayor profundidad los algoritmos concretos con los que se han realizado las pruebas comparativas de nuestro algoritmo. Esto es, el algoritmo K-means y el algoritmo FCM.

Del mismo modo se presentan algunas de las medidas de evaluación utilizadas para medir la validez de una solución de clúster, y que se han utilizado al llevar a cabo las pruebas.

2.1.-Tipos de algoritmos:

2.1.1. Clúster particional

Se basa en trabajar constantemente con un número de particiones del dataset prefijadas de antemano, que se va precisando conforme avanzan las iteraciones hasta lograr obtener la partición que mejor se adapte a los datos.

En cada iteración se escoge un número k de centroides (que no tienen por qué ser elementos del dataset), y se asigna cada elemento exclusivamente a una de las particiones, en función de la distancia con los mismos. Los centroides van desplazándose en base a una función de optimización que busca generar los clústeres que maximicen la distancia interclúster y minimicen la distancia intraclúster.

Los algoritmos de este tipo ofrecen muy buenos resultados al trabajar con datasets hiperesféricos, pero funcionan mal en el resto de casos. También tienden a ofrecer particiones incorrectas cuando existe una gran descompensación entre el número de elementos de las distintas clases, además de requerir conocer a priori el número de clústeres que se han de generar.

No obstante, su alto rendimiento, su sencillo funcionamiento y su utilidad en aplicaciones reales, hacen de ellos algunos de los algoritmos más extendidos y utilizados en la actualidad.

Los algoritmos por excelencia de este tipo son el "K-means" y sus distintas variantes, como puede ser el algoritmo K-means++.

2.1.2. Clúster jerárquico

Existen dos acercamientos completamente opuestos a este tipo de estrategia.

El primero, que trabaja en modo de abajo hacia arriba y se conoce como "aglomerativo", consiste en ver cada ejemplo, inicialmente, como su propio clúster. Con cada iteración se comprueba la distancia entre todos los pares de elementos y se combina uno de ellos, en base a un criterio definido por el "enlazado" escogido (los dos elementos más cercanos, los dos más alejados…). Ejemplos de este tipo de algoritmo son los algoritmos "single-link" o "complete-link".

El segundo es el "divisivo". En este caso, al comienzo de la ejecución, todos los elementos forman un único clúster. Iterativamente, se van separando conjuntos de elementos hasta alcanzar una condición de parada o se alcanza un estado en el que cada ejemplo constituye su propio clúster. Dado el elevado número de posibles particiones que se pueden realizar a partir del clúster inicial, se suele combinar con otros métodos de clúster (como el K-means) para generar estas divisiones.

Ambos algoritmos pueden visualizarse utilizando dendrogramas, lo que permite comprobar la forma en que se han ido generando los clústeres y puede facilitar la elección del número óptimo a escoger.

2.1.3. Clúster basado en densidad

Tiene en cuenta nociones de densidad a la hora de escoger los elementos que pertenecen a cada clúster, buscando generar clústeres en las zonas más densas.

Un punto a favor de este tipo de técnicas es que permiten detectar ruido, definido como los elementos alejados de las zonas más densas en las que se concentran los clústeres. No obstante, su principal problema es que no ofrecen resultados válidos cuando los distintos clústeres se encuentran solapados, situación en la cual estos algoritmos tienen dificultades para definir las fronteras de los mismos.

El algoritmo más conocido de este tipo es el DBSCAN (clúster espacial basado en densidad de aplicaciones con ruido, por sus siglas en inglés), que actúa de manera similar al algoritmo "single-link".

2.1.4. Clúster basado en distribución

Similar al modelo anterior, en este caso el algoritmo busca usar leyes estadísticas para diferenciar los distintos clústeres de un dataset, asumiendo que cada uno de ellos sigue cierta distribución de probabilidad.

Uno de los problemas de este tipo de modelos es el del sobreajuste a los datos (overfit). El problema fundamental, sin embargo, es que, a pesar de que cuenta con una gran base teórica, el hecho de exigir que los datos sigan una función de probabilidad concreta es un requisito muy restrictivo que a menudo no se cumple en datasets reales.

Uno de los algoritmos más conocidos dentro de los de este tipo es el algoritmo EM (esperanza-maximización).

2.1.5. Clúster espectral

Trata el problema de clúster como un problema de particionado de un grafo, generado al enlazar cada pareja de elementos del dataset que cumplan cierta condición relativa a su similitud. Dicho grafo se representa por medio de una matriz de similitud, y la idea final es utilizar los vectores propios de dicha matriz para reducir el espacio dimensional del dataset sobre el que se aplica, facilitando la labor del proceso de clúster. A menudo se aplica el K-means sobre el espacio dimensional reducido, aunque se pueden usar otros algoritmos.

Ofrecen un buen funcionamiento sobre datasets de formas particulares y espacios dimensionales grandes. Además, a diferencia de las dos técnicas anteriores, no requiere ninguna suposición sobre los datos a tratar para resultar efectivo.

2.1.6. Clúster difuso

Usan técnicas similares a los modelos anteriores, aunque en esta ocasión se aprovechan del concepto de variable difusa para asignar a cada elemento, en lugar de un único clúster, un grado de pertenencia a cada uno de los existentes. La idea detrás de este comportamiento está en modelar de algún modo las situaciones en las que es complicado asignar un clúster concreto a un ejemplo determinado, por ejemplo cuando se sitúa en la frontera entre dos clases. De esta manera, además, se modela otro hecho que el resto de algoritmos no tienen en cuenta: que los elementos de distintos clústeres puedan ser similares en parte.

Existen distintos algoritmos que juegan con esta idea, como son el FCM (Fuzzy c-Means) o el PCM (Probabilistic c-Means).

2.1.7. Otros

Pese a la gran cantidad de tipos de algoritmos de clúster que se han presentado, existen muchos otros, algunos de los cuáles resulta complicado encasillar en una de las clases principales por lo peculiar de su acercamiento. Algunos de estos algoritmos pueden ser los bio-inspirados, como es el caso del algoritmo CSO (Chicken Swarm Optimization) [11]; algoritmos genéticos, utilizados para hallar los centroides de una serie de clústeres; o el propio algoritmo gravitacional que trataremos más adelante.

Existen, en particular, otras versiones del algoritmo gravitacional distintas a la propuesta por Wright, que merecen ser presentadas para observar en qué puntos difieren de esta:

Kundu presentó en [12] un modelo similar que utiliza una versión ligeramente distinta de la ley de gravitación de Newton, en la que ignora el producto entre las masas de las partículas y añade un exponente a la distancia entre las mismas. Pone especial énfasis en tener en cuenta, tanto la distancia entre partículas, como la dirección de las rectas que las une. Su modelo permite incorporar nuevas partículas a los clústeres sin tener que repetir todo el proceso desde cero.

En 2007 aparece otra técnica de mano de Blekas y Lagaris [13], aunque no se trata tanto de un algoritmo de clúster, como de un método de preprocesamiento que busca determinar el número de clústeres de un dataset, junto con una estimación de la posición de sus centroides, basándose en leyes de probabilidad.

Por último, en [14] se presenta un método similar al gravitacional, que busca corregir el problema de “agujero negro” en que pueden incurrir los clústeres con mayor masa, sustituyendo el modelo por uno basado en dinámica molecular.

2.2.-Algoritmo K-means

Quizá el algoritmo más extendido y utilizado desde su aparición, el algoritmo K-means se encasilla dentro de los algoritmos particionales. Es un algoritmo que, a pesar de contar con ciertas limitaciones, como son su mala escalabilidad en espacios multi-dimensiones, la necesidad de prefijar el número de clústeres a generar, su incapacidad para tratar con datasets de forma no hiper-esférica o los problemas que tiene al trabajar con cantidades de ejemplos desbalanceadas, ofrece buenos resultados experimentales en gran número de aplicaciones reales. Eso, combinado con su ligereza computacional y su simplicidad y antigüedad, puede explicar el porqué de su éxito.

Dados n ejemplos y un entero k, el algoritmo funciona del siguiente modo:

1. Se generan k centroides aleatoriamente.
2. Se calcula la distancia de cada uno de los n ejemplos a cada uno de los centroides.
3. Se asigna a cada ejemplo el clúster con el centroide más cercano.
4. Se recalcula cada centroide como la media de los ejemplos asignados a su clúster.
5. Se repiten los pasos 2-4 hasta que el algoritmo converge.

Como puede intuirse, el hecho de que la inicialización de los centroides sea aleatoria supone que el algoritmo puede converger a mínimos locales que acarreen una incorrecta clasificación de los ejemplos. Es por ello que se tiende a repetir el algoritmo varias veces para escoger el mejor resultado. Existen variaciones a la hora de inicializar los centroides que tratan de corregir este problema, como sucede con el algoritmo FCM++, pero el resto de su funcionamiento es idéntico [15].

2.3.-Algoritmo FCM (Fuzzy c-Means)

Actúa como una adaptación de la idea del K-means a conjuntos difusos. En este caso, cada elemento viene definido, tras la ejecución del clúster, por un coeficiente de pertenencia a cada uno de los c clústeres generados. La idea es que, de este modo, se puede modelar más fielmente el hecho de que en ocasiones la pertenencia de un ejemplo a un clúster concreto no es absolutamente clara, sino que puede encontrarse cerca de varios clústeres distintos y por tanto pertenecer parcialmente a cada uno de ellos.

El algoritmo resulta similar al anterior, con la salvedad de que, en este caso, en lugar de contar con un único elemento que identifique el clúster al que pertenece un elemento, cada ejemplo llevará asociado un vector de c dimensiones, por lo que se trabajará con una matriz U de dimensión $(n \times c)$, tal que u_{ij} representa la pertenencia al clúster j del ejemplo i. El cálculo de los centroides de cada clúster también habrá de ser diferente, pues tendrá que tener en cuenta la pertenencia de todos los elementos del dataset.

El algoritmo, obviando las fórmulas concretas del mismo, es el siguiente:

1. Se fijan una serie de parámetros iniciales c (número de clústeres), m, $LMAX$ (máximo de iteraciones), la función A y la A-norma $\|k\|_A$. Se escoge una matriz inicial $U^{(0)} \in M_{fc} \equiv$ conjunto de las particiones difusas posibles. En cada paso k, $k = 0, 1, \dots, LMAX$:
2. Se computan los centroides de cada clúster a partir de la matriz de pertenencia U.
3. Se genera una matriz de pertenencia actualizada $\widehat{U}^{(k+1)}$.

4. Se compara $\widehat{U}^{(k+1)}$ con $\widehat{U}^{(k)}$, utilizando cualquier norma matricial conveniente. Si $\left\|\widehat{U}^{(k+1)} - \widehat{U}^{(k)}\right\| <\in$, el algoritmo finaliza. En caso contrario se fija $\widehat{U}^{(k)} = \widehat{U}^{(k+1)}$ y se repiten los pasos 2-4.

Se ha obviado premeditadamente explicar el significado de algunos parámetros y funciones por encontrarse fuera del objetivo de este trabajo. Basta con saber que el algoritmo requiere utilizar la A-norma $\|k\|_A$ para computar la distancia de cada elemento a cada centroide, que el parámetro m es un exponente que determina lo duro o difuso de los clústeres, y que es necesario calcular los elementos de la matriz de pertenencia. Se puede encontrar una explicación en profundidad en [16].

2.4.-Medidas de evaluación

Existen dos acercamientos al problema de evaluar el resultado de un método de clústering [9] [17]:

2.4.1. Evaluación interna:

Dado que los algoritmos de clústering forman parte de los métodos de aprendizaje no supervisado, en la mayoría de casos no se dispone de una clasificación a priori de los datos con la que estos puedan compararse, a diferencia de como sucede en la clasificación supervisada. Es por ello que a menudo se recurre a un tipo de evaluación, llamada interna, que se limita a evaluar la estructura de los clústeres generados como salida, sin tener en cuenta ningún tipo de conocimiento previo.

La idea detrás de la evaluación interna consiste en calcular la similitud entre los miembros de cada clúster, así como la disimilitud entre clústeres, de modo que se premien las agrupaciones que formen conjuntos más uniformes.

Las medidas empleadas para la evaluación del algoritmo son las siguientes:

- **Índice Davies-Bouldin**

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i,\ c_j)}\right)$$

n ≡ Número de clústeres
c_x ≡ Centroide del cluster x
σ_x ≡ Distancia media de todos los elementos del clúster x a c_x
$d(c_i, c_j)$ ≡ Distancia entre los centroides c_i y c_j

Mide, en promedio, el ratio entre la similitud intraclúster y la disimilitud interclúster de cada clúster y aquel del que está más cerca. Como el objetivo es obtener clústeres compuestos por elementos similares y diferentes del resto de clústeres, a menor índice Davies-Bouldin mejor resultado.

- **Índice Dunn**

Se define como el ratio entre la mínima distancia inter-clúster y la máxima intra-clúster del sistema:

$$D = \frac{\min\limits_{1\leq i<j\leq n} d(i,j)}{\max\limits_{1\leq k\leq n} d'(k)}$$

$d(c_i, c_j)$ ≡ Distancia entre los clústeres i y j (distancia entre los centroides)
d'(k) ≡ Distancia intraclúster de k (distancia entre miembros más alejados del clúster k)

A diferencia del índice Davies-Bouldin, un mayor valor de D representa un mejor clústering. Es importante observar que el índice Dunn es una medida del caso peor, dado que emplea la distancia interclúster entre los clústeres más cercanos y la intraclúster del clúster más amplio.

2.4.2. **Evaluación externa:**

Por contrapartida existe la evaluación externa. La componen aquellos métodos de evaluación que utilizan algún tipo de información no empleada a la hora de realizar el clústering, como puede ser la clasificación a priori de los ejemplos, o benchmarks externos. Resultan un modo de aproximación similar al empleado para las tareas de clasificación, con la salvedad de que la mayoría de estos métodos cuentan, en lugar del número de datos clasificados correctamente, el número de parejas de elementos predichos en una misma clase, que acaban estándolo.

Las medidas empleadas para la evaluación del algoritmo son las siguientes:

- **Pureza (Purity)**

Dados un conjunto de clústeres M y uno de clases D, ambos clasificando N ejemplos:

$$Pureza = \frac{1}{N} \sum_{m \in M} \max_{d \in D} |m \cap d|$$

|x| ≡ Cardinal del conjunto x

Busca medir el grado en que cada clúster contiene una única clase.

Problema: No penaliza un número elevado de clústeres. Se obtiene una pureza de 1 (máxima) cuando cada clúster está compuesto por un único ejemplo.

- **Medida Rand (Índice Rand):**

Calcula la similitud de un clústering con las clasificaciones conocidas a priori. Puede interpretarse como el ratio de decisiones correctas obtenidas por el algoritmo:

$$RI = \frac{TP + FN}{TP + FP + TN + FN} = \frac{TP + FN}{\binom{n}{2}}$$

TP ≡ Verdaderos positivos

TN ≡ Verdaderos negativos

FP ≡ Falsos positivos

FN ≡ Falsos negativos

n ≡ Número de ejemplos

Se analizan todas las parejas de ejemplos del sistema. Se cuenta como positivo cualquier pareja de elementos que se encuentre en un mismo clúster. Dicho positivo será verdadero o falso en función de si esa misma pareja de elementos compartía clase en la clasificación conocida, o no.

Problema: Da la misma importancia a falsos positivos que a falsos negativos.

- **F-measure:**

Evalúa tanto la precisión, como la exhaustividad (recall) del clústering obtenido:

$$F_1 = 2\,\frac{precision * recall}{precision + recall}$$

$$precision\ = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Se calcula como la media harmónica entre la precisión y la exhaustividad.

Problema: No tiene en cuenta los falsos positivos.

Nota: Aunque se han tomado medidas usando todos los métodos anteriores, se obviarán algunos resultados redundantes que no aportan nueva información.

3.-PRELIMINARES

3.1.-Funciones de agregación

Definición 3.1. Una función de agregación n-dimensional es una función $M: [0,1]^n \to [0,1]$ que verifica las dos condiciones siguientes:

- M es no decreciente en cada una de sus variables;
- $M(0,\dots,0) = 0$ y $M(1,\dots,1) = 1$.

Ejemplos: Medias, producto, mínimo, máximo…

Definición 3.2. Sea $M: [0,1]^n \to [0,1]$ una función de agregación *n*-dimensional.

(i). $a \in [0,1]$ es un aniquilador de M si $M(x_1,\dots,x_n) = a$ siempre que $a \in \{x_1,\dots,x_n\}$.
(ii). Si M no tiene aniquilador, M se dice estrictamente creciente si lo es en el dominio $[0,1]^n$ como función real de n variables. Si a es un aniquilador de M, si es estrictamente creciente en el dominio $([0,1]\backslash\{a\})^n$.
(iii). M tiene divisores de cero si existen $x_1,\dots,x_n \in [0,1]$ tales que $M(x_1,\dots,x_n) = 0$.
(iv). M es idempotente si $M(x_1,\dots,x_n) = x$ para todo $x \in [0,1]$.

Nos vamos a centrar en el caso de funciones de agregación en dimensión 2. Por ello, revisamos algunas propiedades que nos van a resultar de interés.

Definición 3.3. Sea M una función de agregación de dos variables.

(i). M es simétrica si $M(x,y) = M(y,x)$ para cualesquiera $x,y \in [0,1]$.
(ii). M es asociativa si $M(M(x,y),z) = M(x,M(y,z))$ para cualesquiera $x,y,z \in [0,1]$.

La asociatividad permite extender las funciones de agregación bidimensionales a dimensiones mayores de una manera "razonable". En particular, es una de las propiedades fundamentales que se demandan en la construcción de normas triangulares (t-normas).

Definición 3.4. Una norma triangular o t-norma es una función de agregación $T: [0,1]^2 \to [0,1]$ asociativa y simétrica tal que $T(1,x) = x$ para todo $x \in [0,1]$.

Las t-normas permiten modelar intersecciones entre conjuntos difusos. Ahora bien, si estamos trabajando con solo dos elementos, la asociatividad no es en principio una propiedad que deba ser requerida. Esta idea, la de poder representar la intersección sin necesariamente requerir la asociatividad nos lleva al concepto de función de solapamiento (función de overlap) que estudiamos a continuación.

3.2.-Funciones de overlap

En esta subsección presentamos fundamentalmente los resultados en [18].

Definición 3.5. Una función $O: [0,1]^2 \to [0,1]$ es una función de solapamiento si

(O1). O es simétrica;
(O2). $O(x,y) = 0$ si y solo si $xy = 0$;
(O3). $O(x,y) = 1$ si y solo si $xy = 1$;
(O4). O es no decreciente;
(O5). O es continua.

Los siguientes son dos ejemplos de funciones de solapamiento.

- $O(x,y) = \min(x,y)$
- $O(x,y) = xy$

Estas dos funciones son en particular, t-normas. De hecho, toda t-norma continua sin divisores de cero (es decir, tal que $T(x,y) = 0$ implica que $xy = 0$) es una función de overlap. Sin embargo, existen funciones de overlap que no son funciones de solapamiento, como por ejemplo:

$$O(x,y) = \min\left(xy^k, x^k y\right) \text{ con } k > 0$$

que no es una t-norma si $k \neq 1$,

$$O(x,y) = (xy)^p \text{ con } p > 0$$

que no es una t-norma si $p \neq 1$, o

$$O(x,y) = \sin\frac{\pi}{2}(xy)^p \text{ con } p > 0$$

que no es nunca una t-norma.

Podemos, en particular, presentar el siguiente resultado sobre la relación entre funciones de solapamientos y t-normas.

Teorema 3.6. *Sea O una función de solapamiento asociativa. Entonces O es una t-norma.*

En cuanto a la construcción de funciones de overlap, es posible hacerlo de forma general tal y como muestra el siguiente resultado.

Teorema 3.7. *La función $O\colon [0,1]^2 \to [0,1]$ es una función de solapamiento si y solo si*

$$O(x,y) = \frac{f(x,y)}{f(x,y) + h(x,y)}$$

Con $f, h\colon [0,1]^2 \to [0,1]$ tales que

1) *f y h son simétricas;*
2) *f es no decreciente y h es no creciente;*
3) *$f(x,y) = 0$ si y solo si $xy = 0$;*
4) *$h(x,y) = 0$ si y solo si $xy = 1$;*
5) *f y h son continuas;*

Por ejemplo, si $f(x,y) = \sqrt{xy}$ y $h(x,y) = \max(1-x, 1-y)$, entonces obtenemos la función de solapamiento

$$O(x,y) = \frac{\sqrt{xy}}{\sqrt{xy} + \max(1-x, 1-y)}$$

3.3.- El algoritmo de clúster gravitacional

El algoritmo de clúster gravitacional pretende hacer uso de la Ley de Gravitación Universal de Newton para llevar a cabo un proceso de clasificación no supervisado.

El esquema del algoritmo, tal y como figura en [5], es el siguiente:

Supongamos que tenemos n partículas $p_1, \dots, p_n$, para las que conocemos sus posiciones $s_1, \dots, s_n \in \mathbb{R}^n$.

1. Asignamos a cada partícula p_i una masa 1 para comenzar el algoritmo.
2. Fijamos dos parámetros reales positivos δ y ϵ.
   - Utilizamos δ para determinar la longitud de cada paso temporal dt. En concreto, en el intervalo $[t, t+dt]$ la partícula que se mueva más deprisa debe desplazarse una distancia δ.
   - Si dos partículas están en algún momento a distancia menor que ϵ, las unimos en una partícula con masa igual a la suma de las masas original y posición dada por el centro de masas de las partículas originales.
3. Inicializamos el tiempo con $t = 0$.
4. Repetimos los pasos (i)-(iv) siguientes hasta que solo quede una partícula.
   (i). En cada intervalo de tiempo $[t, t+dt]$, para cada partícula i calculamos el efecto gravitacional de las demás partículas sobre ella. En concreto:

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} \frac{m_i(t) m_j(t)}{m_i(t)} \frac{s_j(t) - s_i(t)}{|s_j(t) - s_i(t)|} \frac{1}{|s_j(t) - s_i(t)|^2} dt^2 \qquad (1)$$

   donde G es una constante positiva.

   (ii). Para cada partícula i, su nueva posición es:

$$s_i(t+dt) = s_i(t) + g(i,t,dt)$$

   (iii). Incrementamos el tiempo de t a $t+dt$.

   (iv). Si dos partículas i y j están a distancia (euclídea) menor que ϵ, se funden en una partícula como se ha explicado anteriormente.

Al final tenemos una sola partícula y, si el proceso ha durado un tiempo T, la siguiente evolución:

- De $t_n = 0$ *a* t_{n-1} había n partículas.
- De t_{n-1} *a* t_{n-2} había $n-1$ partículas.
- …
- De t_3 *a* t_2 había 3 partículas.
- De t_2 *a* t_1 había 2 partículas.

Definimos la vida relativa de la configuración de k clusters como

$$R_k = \frac{t_k - t_{k-1}}{T}$$

Tomamos como solución aquella configuración R_{k_0} con la mayor vida relativa.

Nótese que en una iteración dada puede haber varias fusiones de partículas. El caso en que en t_k se tenían k partículas y en t_{k-1} pasa a haber $k-2$ partículas, se puede ver como que de t_k a t_{k-1} había k partículas y de t_{k-1} a t_{k-2} había $k-1$ partículas. Dado que $t_{k-1} = t_{k-2} \rightarrow R_{k-1} = 0$.

Este modelo puede generalizarse reemplazando la Ecuación (1) por

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} \frac{m_i(t)^p m_j(t)^q}{m_i(t)} \frac{s_j(t) - s_i(t)}{|s_j(t) - s_i(t)|} \frac{1}{|s_j(t) - s_i(t)|^2} dt^2 \qquad (2)$$

con $p, q \geq 0$. En particular, los mejores resultados se obtienen cuando $p = q = 0$, obteniéndose el denominado modelo unitario de Markov

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} \frac{1}{m_i(t)} \frac{s_j(t) - s_i(t)}{|s_j(t) - s_i(t)|} \frac{1}{|s_j(t) - s_i(t)|^2} dt^2 \qquad (3)$$

4.-UN NUEVO ALGORITMO DE CLÚSTER GRAVITACIONAL

En esta sección presentamos el contenido principal del trabajo: un nuevo algoritmo de clúster gravitacional que generaliza el mostrado en la sección [2.4], reemplazando el factor $\frac{1}{m_i(t)}$ en la ecuación (4) por un término de la forma $O\left(1,\frac{1}{m_i(t)}\right)$ donde $O\colon [0,1]^2 \to [0,1]$ es una función de overlap general. Ahora bien, dado que la función de overlap está definida sobre el cuadrado unidad $[0,1]^2$, en principio es necesaria una normalización. Esto puede hacerse considerando la función

$$O\left(\frac{m_i^{p-1}}{n^{p-1}},\frac{m_i^q}{n^q}\right)$$

Con $p, q > 0$. Sin embargo, dado que estamos interesados en el caso p, q próximos a cero (que es el que mejor funciona en el trabajo original), se tiene que esta función siempre es mayor o igual que $O\left(1,\frac{1}{m_i(t)}\right)$. Por ello, sea $O\colon [0,1]^2 \to [0,1]$ una función de overlap. Dada una configuración de n partículas $p_1, \dots, p_n$, para las que conocemos sus posiciones $s_1, \dots, s_n \in \mathbb{R}^n$, el algoritmo que proponemos es el siguiente:

1. Asignamos a cada partícula p_i una masa 1 para comenzar el algoritmo.
2. Fijamos dos parámetros reales positivos δ y ϵ.
    - Utilizamos δ para determinar la longitud de cada paso temporal dt. En concreto, en el intervalo $[t, t+dt]$ la partícula que se mueva más deprisa debe desplazarse una distancia δ.
    - Si dos partículas están en algún momento a distancia menor que ϵ, las unimos en una partícula con masa igual a la suma de las masas original y posición dada por el centro de masas de las partículas originales.
3. Inicializamos el tiempo con $t = 0$.
4. Repetimos los pasos (i)-(iv) siguientes hasta que solo quede una partícula.
    (i). En cada intervalo de tiempo $[t, t+dt]$, para cada partícula i calculamos el efecto gravitacional de las demás partículas sobre ella. En concreto:

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} O\left(1,\frac{1}{m_i(t)}\right) \frac{s_j(t) - s_i(t)}{|s_j(t) - s_i(t)|} \frac{1}{|s_j(t) - s_i(t)|^2} dt^2 \qquad (4)$$

    donde G es una constante positiva.

    (ii). Para cada partícula i, su nueva posición es:

$$s_i(t+dt) = s_i(t) + g(i,t,dt)$$

    (iii). Incrementamos el tiempo de t a $t + dt$.
    (iv). Si dos partículas i y j están a distancia (euclídea) menor que ϵ, se funden en una partícula como se ha explicado anteriormente.

Al final tenemos una sola partícula y, si el proceso ha durado un tiempo T, la siguiente evolución:

- De $t_n = 0$ *a* t_{n-1} había n partículas.
- De t_{n-1} *a* t_{n-2} había $n-1$ partículas.

- …
- De t_3 a t_2 había 3 partículas.
- De t_2 a t_1 había 2 partículas.

Definimos la vida relativa de la configuración de k clusters como

$$R_k = \frac{t_k - t_{k-1}}{T}$$

Tomamos como solución aquella configuración R_{k_0} con la mayor vida relativa.

Nótese que en una iteración dada puede haber varias fusiones de partículas. El caso en que en t_k se tenían k partículas y en t_{k-1} pasa a haber $k-2$ partículas, se puede ver como que de t_k a t_{k-1} había k partículas y de t_{k-1} a t_{k-2} había $k-1$ partículas. Dado que $t_{k-1} = t_{k-2} \rightarrow R_{k-1} = 0$.

5.-EXPERIMENTACIÓN

La evolución del algoritmo pasó por distintas fases, a lo largo de las cuáles se realizaron diversas modificaciones en busca de una mejora significativa respecto al algoritmo original. A continuación se hará una breve recapitulación de las mismas, hasta alcanzar el modelo definitivo, para pasar a describir su comportamiento y comparar sus resultados, tanto con el algoritmo gravitacional clásico, como con el K-means y el FCM.

5.1.-Aproximación inicial: Overlaps sobre masas normalizadas

Inicialmente se trató de adaptar el algoritmo para trabajar con masas normalizadas, de modo que pudiesen aplicarse overlaps comunes. Los overlaps planteados fueron los siguientes:

Overlap 1 -> Producto:

$$O(x,y) = x * y \rightarrow O\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^{q}}{n^{q}}\right) = \frac{{m_i}^{p-1}}{n^{p-1}} * \frac{{m_j}^{q}}{n^{q}}$$

Overlap 2:

$$O(x,y) = \min\left(x^k y, x y^k\right) \rightarrow O\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^{q}}{n^{q}}\right) = \min\left(\left(\frac{{m_i}^{p-1}}{n^{p-1}}\right)^k * \frac{{m_j}^{q}}{n^{q}}, \frac{{m_i}^{p-1}}{n^{p-1}} * \left(\frac{{m_j}^{q}}{n^{q}}\right)^k\right)$$

Overlap 3:

$$O(x,y) = \frac{xy}{1+(1-x)(1-y)} \rightarrow O\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^{q}}{n^{q}}\right) = \frac{\frac{{m_i}^{p-1}}{n^{p-1}} * \frac{{m_j}^{q}}{n^{q}}}{1+\left(1-\frac{{m_i}^{p-1}}{n^{p-1}}\right)\left(1-\frac{{m_j}^{q}}{n^{q}}\right)}$$

Overlap 4:

$$O(x,y) = \frac{\sqrt{xy}}{\sqrt{xy}+\max(1-x,1-y)} \rightarrow O\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^{q}}{n^{q}}\right) = \frac{\sqrt{\frac{{m_i}^{p-1}}{n^{p-1}} * \frac{{m_j}^{q}}{n^{q}}}}{\sqrt{\frac{{m_i}^{p-1}}{n^{p-1}} * \frac{{m_j}^{q}}{n^{q}}}+\max\left(1-\frac{{m_i}^{p-1}}{n^{p-1}}, 1-\frac{{m_j}^{q}}{n^{q}}\right)}$$

Overlap 5 -> Seno:

$$O(x,y) = \sin\left(\frac{\pi}{2}(xy)^k\right) \rightarrow O\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^{q}}{n^{q}}\right) = \sin\left(\frac{\pi}{2}\left(\frac{{m_i}^{p-1}}{n^{p-1}} * \frac{{m_j}^{q}}{n^{q}}\right)^k\right)$$

Overlap 6 -> Mínimo:

$$O(x,y) = \min(x,y)^k \rightarrow O\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^{q}}{n^{q}}\right) = \min\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^{q}}{n^{q}}\right)^k$$

Los resultados obtenidos, no obstante, fueron muy inferiores a los logrados por el modelo unitario de Markov. En la figura 1 se pueden observar los valores de índice-rand devueltos para algunas combinaciones de parámetros utilizando los distintos overlaps sobre masas normalizadas. El resto de combinaciones probadas ofrecieron valores similares, que en ningún caso superaban el valor devuelto por los algoritmos K-means, FCM o el algoritmo clásico.

Además, el hecho de emplear masas normalizadas implica que al elevar una masa a un exponente cada vez mayor, el valor obtenido va disminuyendo. En concreto, cuando los valores de p y q son muy grandes, los valores de las masas elevados a los mismos tienden a 0, lo que neutraliza, no solo la influencia que la propia masa juega en la ecuación, sino también el valor del resto de términos. Es por todo esto que la idea se descartó rápidamente, y se pasaron a utilizar masas sin normalizar.

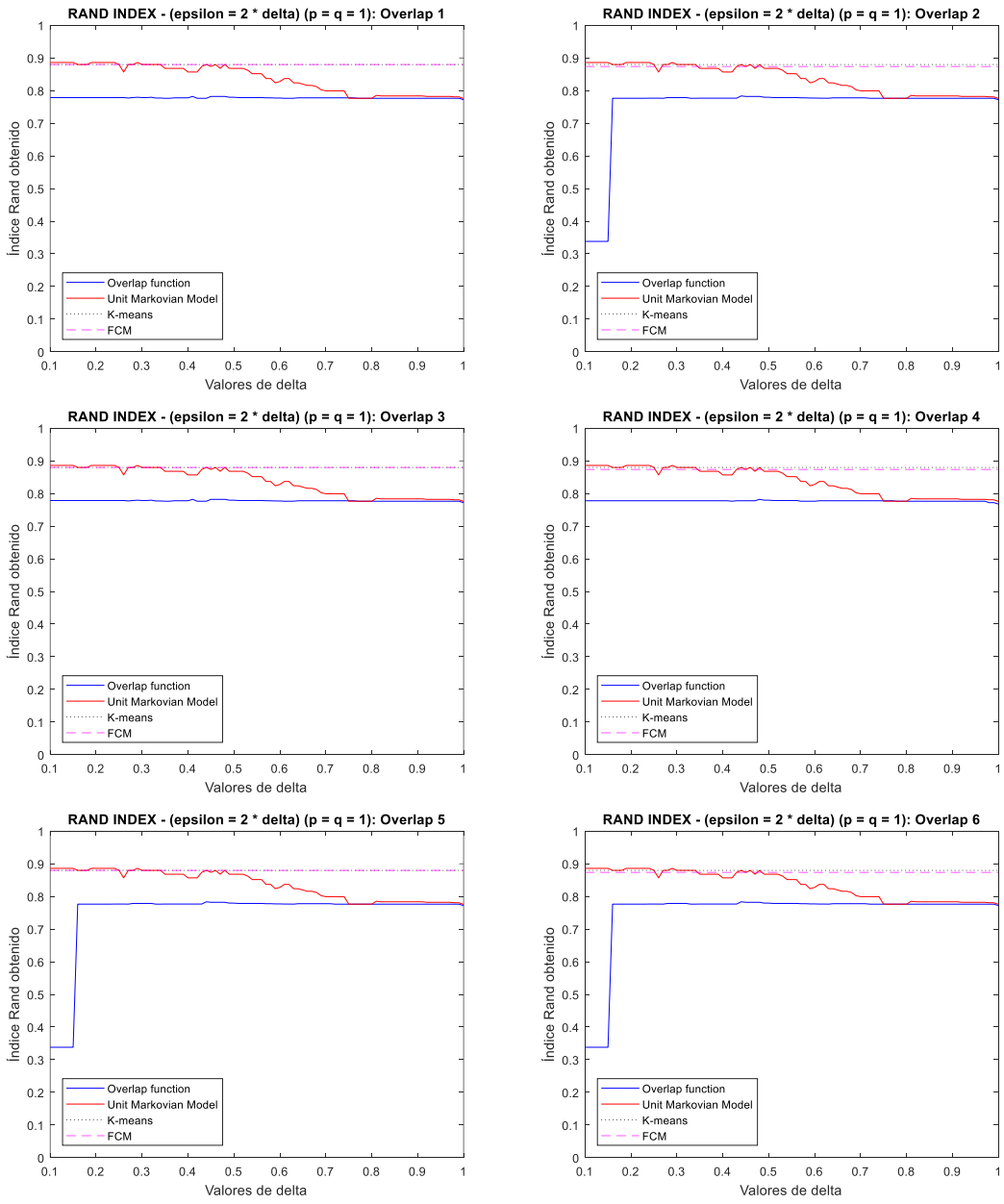


Fig. 1 Índice Rand utilizando overlaps normalizados

5.2.-Pruebas con overlaps sobre masas sin normalizar

Los overlaps empleados para reemplazar al término O de la ecuación (4) fueron los siguientes:

$$1.\quad O(x,y) = (x * y)^p \rightarrow O\left(\frac{1}{m_i}, 1\right) = \frac{1}{{m_i}^p}$$

$$2.\quad O(x,y) = \frac{\sqrt{x}}{\sqrt{x}+\max(1-x,1-y)} \rightarrow O\left(\frac{1}{m_i}, 1\right) = \frac{\sqrt{\frac{1}{m_i}}}{\sqrt{\frac{1}{m_i}}+\max\left(1-\frac{1}{m_i},0\right)}$$

$$3.\quad O(x,y) = \sin(\frac{\pi}{2} * (xy)^p) \rightarrow O\left(\frac{1}{m_i}, 1\right) = \sin\left(\frac{\pi}{2}\left(\frac{1}{m_i}\right)^p\right)$$

Planteamiento de las pruebas

En las pruebas se comparó el resultado obtenido por el algoritmo gravitacional utilizando los métodos de evaluación previamente mencionados con el algoritmo gravitacional clásico, el Kmeans y el FCM. Los resultados tomados vienen determinados por algunas de las medidas descritas en la sección 2.4.

Uno de los primeros problemas que presenta el algoritmo original es la necesidad de seleccionar unos valores adecuados para los parámetros δ y ϵ, que determinan enormemente los resultados del algoritmo. Las pruebas iniciales demostraron una gran variación en las clasificaciones obtenidas para cambios muy pequeños de ambos parámetros, lo que supone un grave problema a la hora de realizar las pruebas.

El método escogido fue tomar un valor de delta inicial pequeño (a menor valor de delta mayor fidelidad al modelo gravitacional real, aunque mayor coste temporal), e ir incrementándolo ligeramente hasta un valor máximo fijado. Aunque se realizaron pruebas utilizando valores de ϵ distintos, finalmente se optó por seguir las indicaciones del artículo original y tomar ϵ = 2δ, valor suficientemente bueno como para asegurar la convergencia del algoritmo.

Del mismo modo, se tomaron distintos valores de $p \geq 0$, con el fin de encontrar el modelo que ofreciese mejores resultados, y las diferencias producidas al aumentar dicho valor. A continuación se realiza un resumen de los resultados obtenidos más relevantes.

Pruebas iniciales:

Inicialmente se testearon los distintos overlaps con varios datasets reales de distinta forma y tamaño, a modo de aproximación inicial para encontrar los más prometedores. Los datasets empleados se tomaron de [19] y [20], y son los siguientes:

- **Dataset Iris**:
  Consta de 150 ejemplos clasificados en 3 clases distintas, la primera de ellas fácilmente diferenciable de las otras dos, que lo son en menor medida. Los ejemplos están compuestos por 4 variables reales de rangos similares.
- **Dataset balance:**
  625 ejemplos compuestos por 4 variables enteras en {1, 2, 3, 4, 5} que representan 3 estados de una balanza: desviada a la izquierda, desviada a la derecha o en equilibrio.
- **Dataset banana**:
  Dataset sintético bidimensional formado por 2 clases difíciles de clasificar, con fronteras muy próximas, dispuestos en forma de banana cuando se representan gráficamente. Compuesto por 5300 ejemplos.

Dataset Iris:

Como puede observarse en las figuras 2-4, para un dataset relativamente sencillo de clasificar, el overlap 1 devuelve resultados mediocres, al igual que el overlap 3. El overlap 2 ofrece unos resultados similares al modelo unitario de Markov, aunque no está afectado por el valor de p y, por tanto, no puede ser mejorado. Con el fin de observar si el resto de overlaps podían mejorar, se probó a incrementar el valor de p.

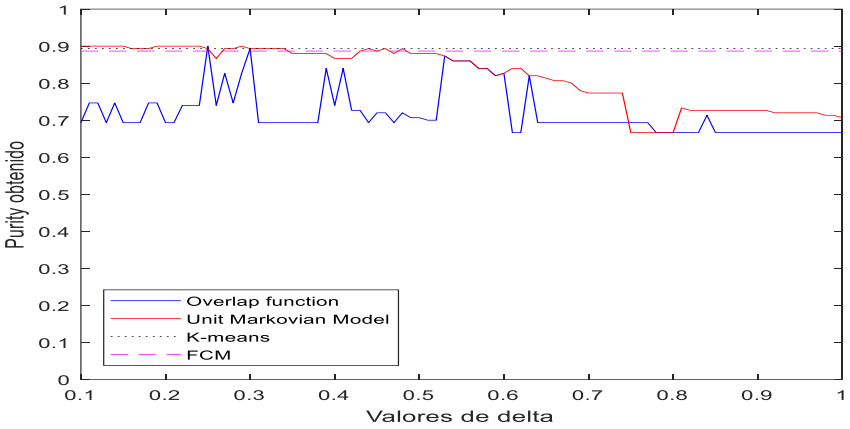


Fig. 2 Overlap 1 - p = 0 (Purity)

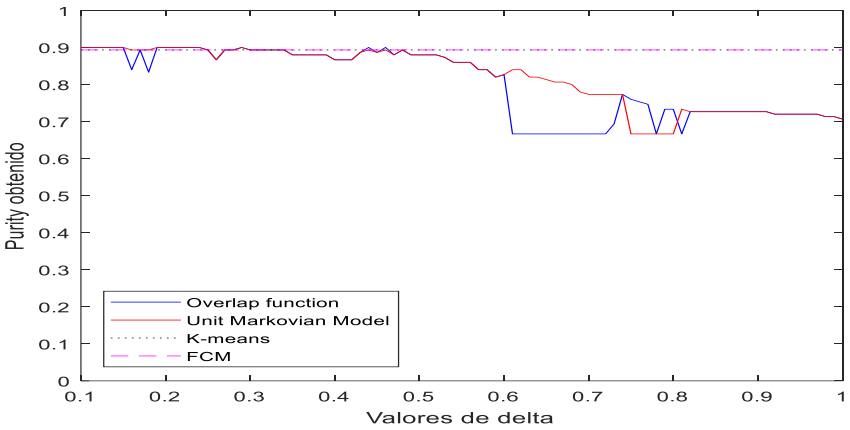


Fig. 3 Overlap 2 (Purity)

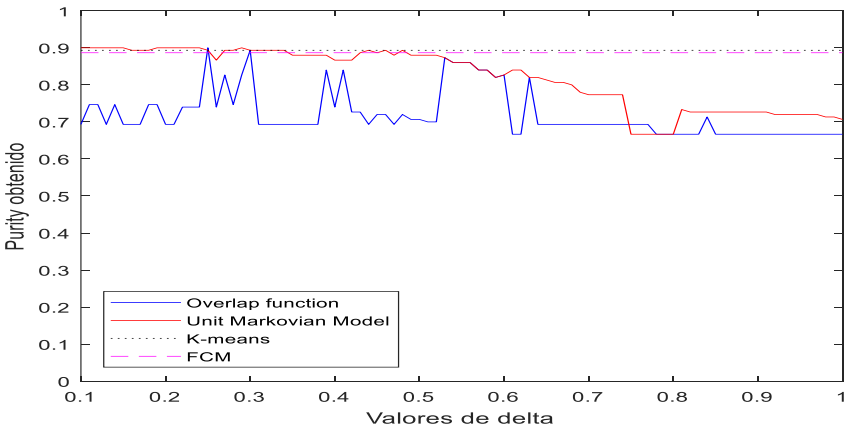


Fig. 4 Overlap 3 - p = 0 (Purity)

Pese a que la respuesta del algoritmo se mantiene por lo general estable para $p > 0$, existen valores de delta para los cuales logra superar el resultado, tanto del modelo unitario de Markov, como de los algoritmos K-means y FCM.

Como curiosidad, cabe destacar que el overlap 3 obtiene los mismos resultados cuando se utilizan valores de p muy grandes, lo cuál tiene sentido pues los valores de ambos overlaps tienden a 0 conforme p aumenta.

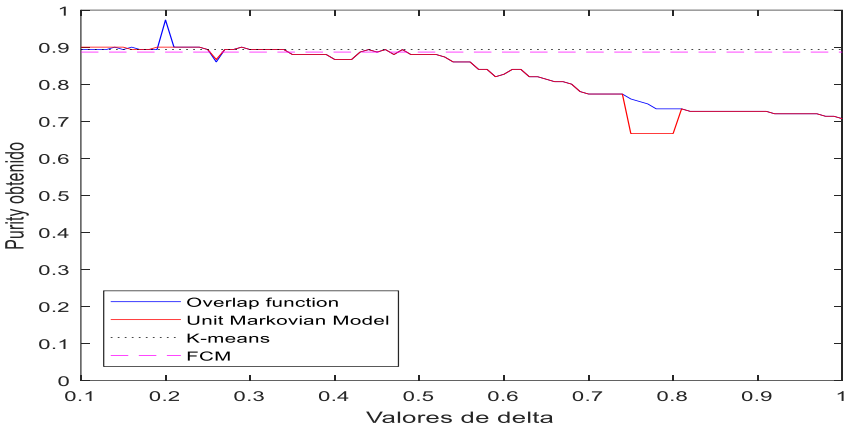


Fig. 5 Overlap 1 - p = 10 (Purity)

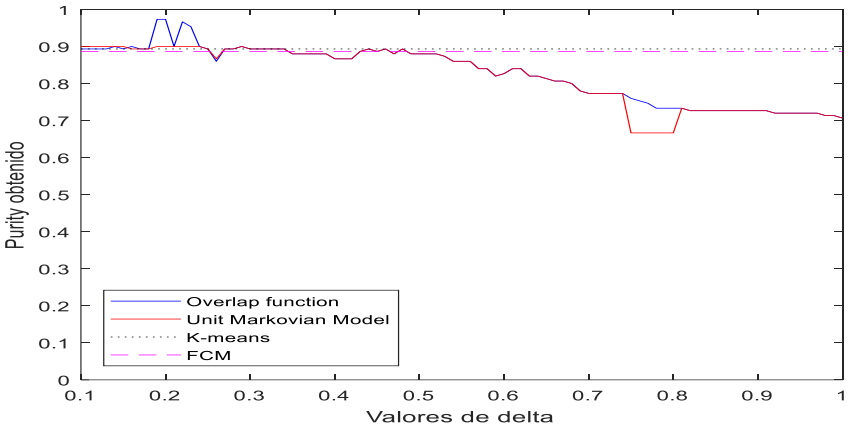


Fig. 6 Overlap 1 - p = 20 (Purity)

Dataset balance:

Aunque este dataset muestra una variabilidad mayor en cuanto a la precisión de los resultados, se sigue percibiendo cierta mejora conforme el valor de p aumenta para el overlap 1, al menos para la pureza y el índice-Rand (sus gráficas son prácticamente idénticas). El F-measure, no obstante, empeora con el aumento de p. Pese a todo, el mejor resultado se obtiene para el modelo unitario de Markov, aunque el overlap 1 obtiene mejores valores en media. El overlap 2 mantiene un comportamiento similar al modelo unitario de Markov.

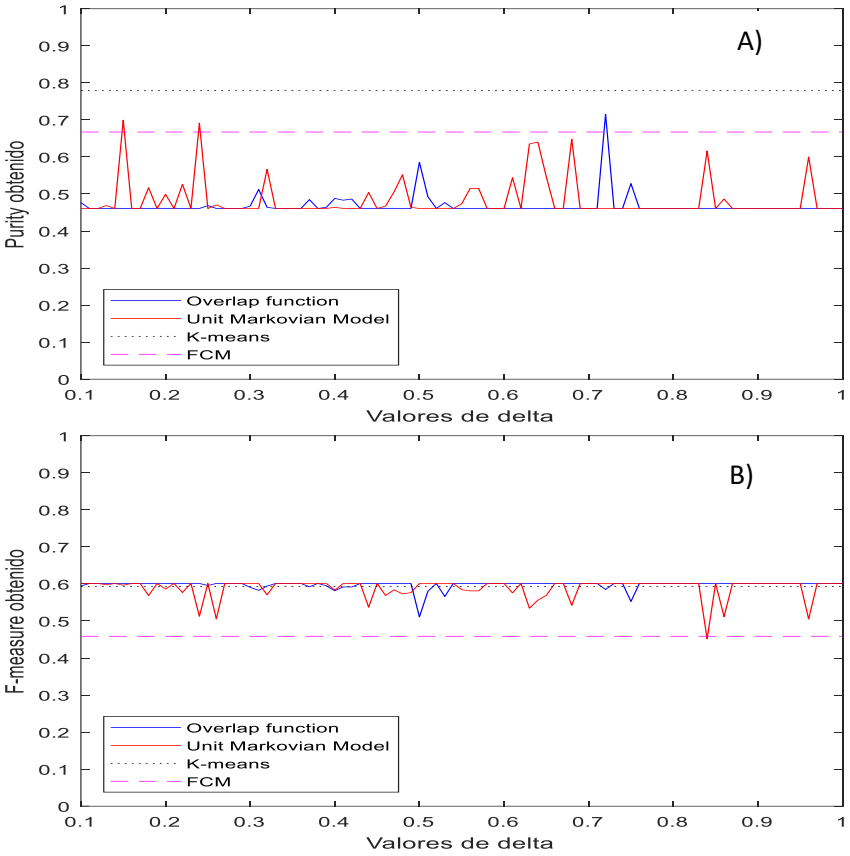


Fig. 7 Overlap 2 (A) Purity, B) F-measure)

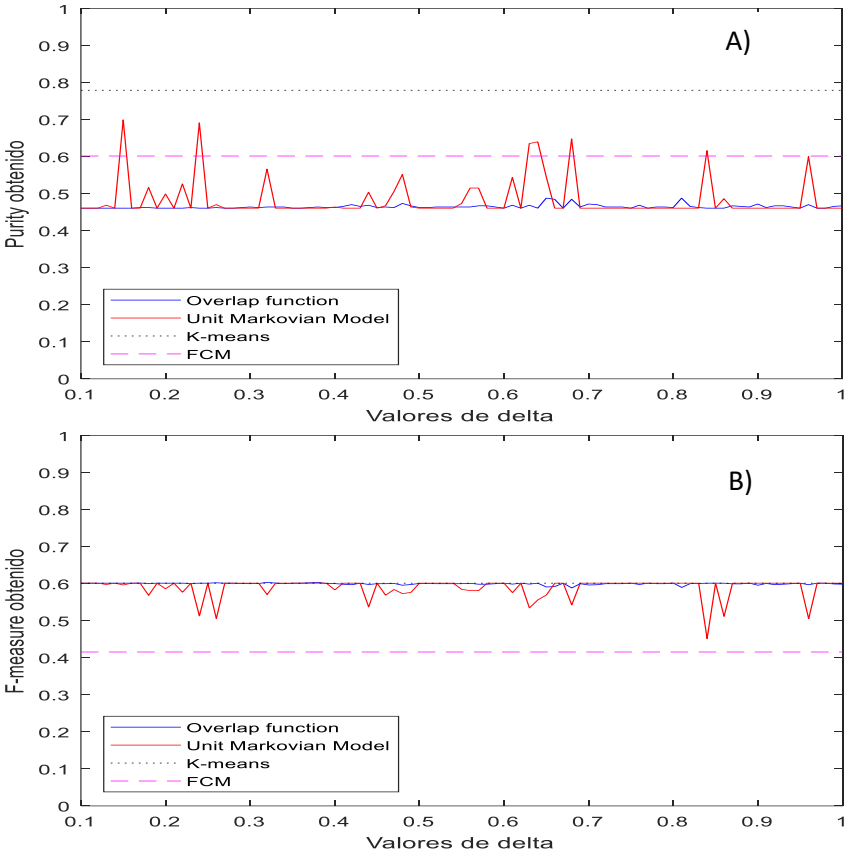


Fig. 8 Overlap 1 - p = 0 (A) Purity, B) F-measure)

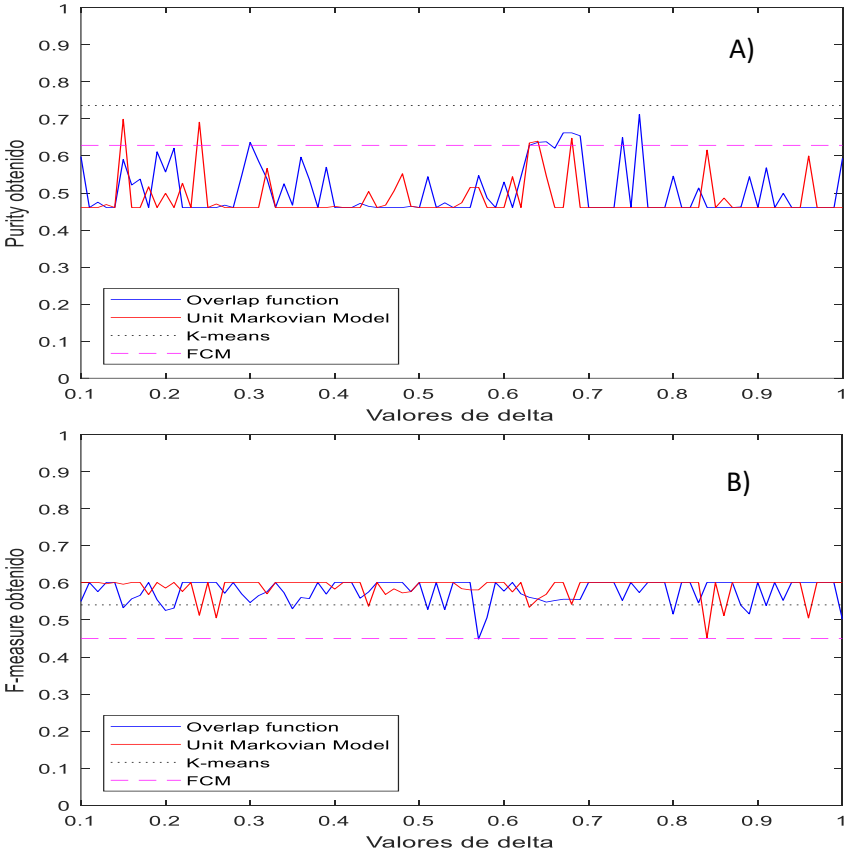


Fig. 9 Overlap 1 – p = 2 (A) Purity, B) F-Measure)

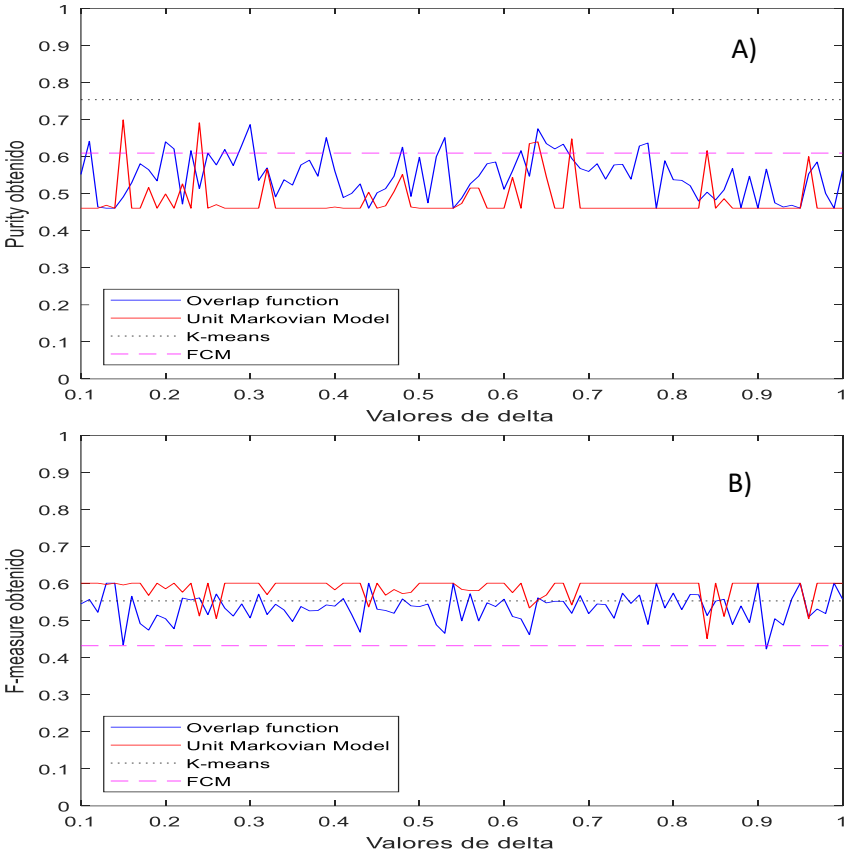


Fig. 10 Overlap 1 - p = 10 (A) Purity, B) F-measure)

Dataset banana:

Una vez más, pese a que los resultados obtenidos por el algoritmo no son excesivamente buenos, los mejores se obtienen para valores elevados de p. Los resultados del overlap 2 son mediocres en este caso.

Como puede comprobarse, tampoco los algoritmos K-means o FCM funcionan bien para los dos últimos datasets, probablemente debido a lo difuso de sus clases.

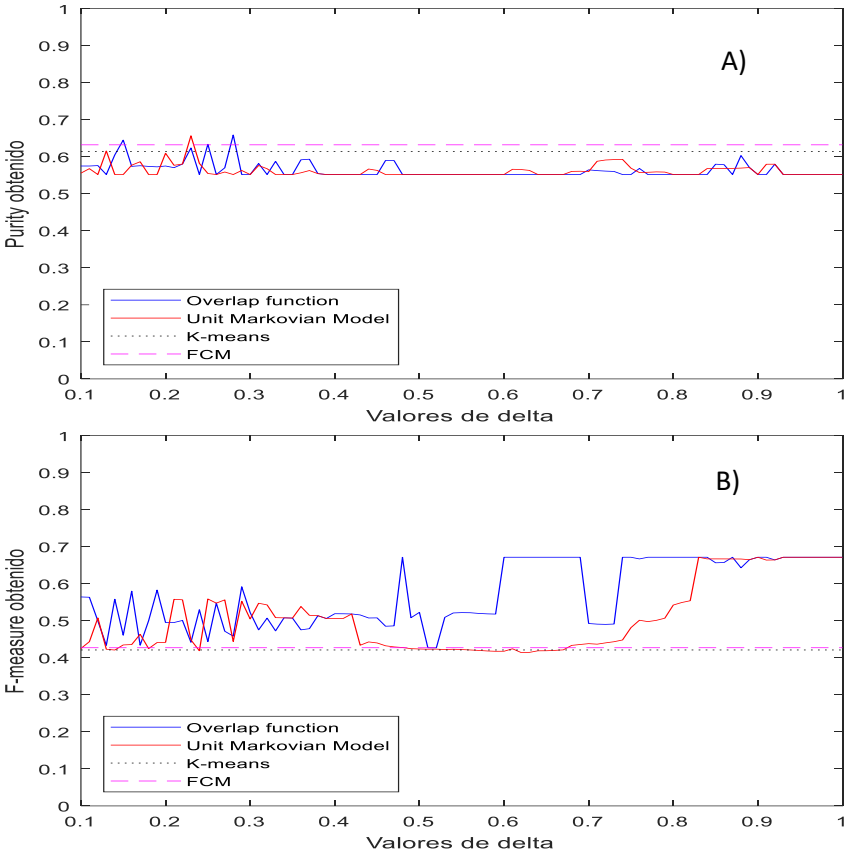


Fig. 11 Overlap 2 (A) Purity, B)F-measure)

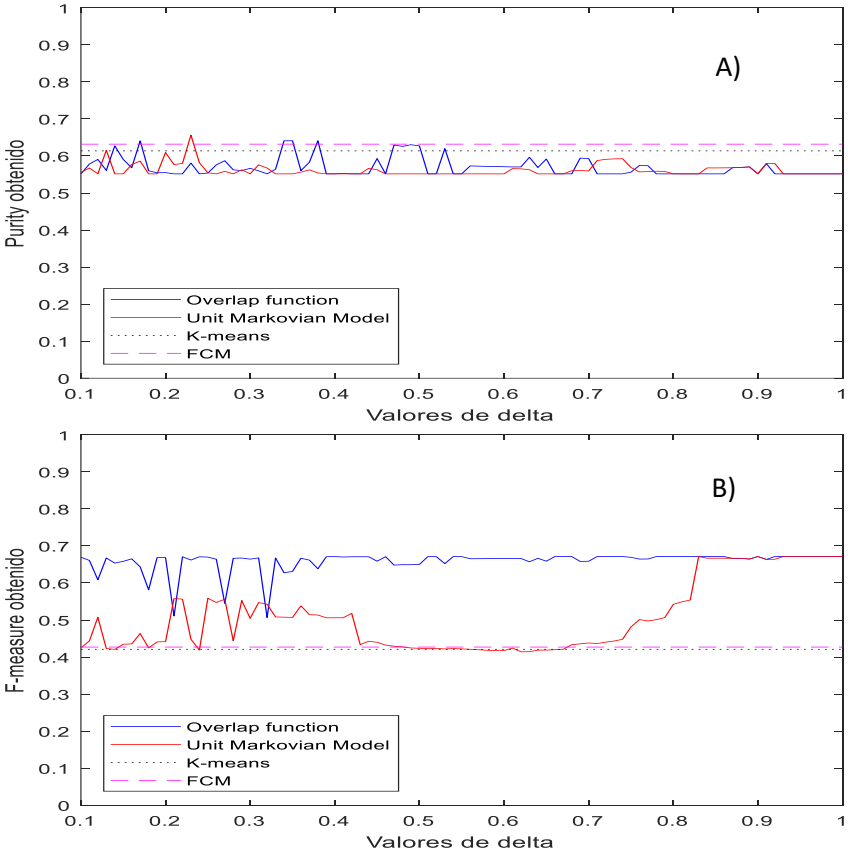


Fig. 12 Overlap 1 - p = 0 (A) Purity, B) F-measure)

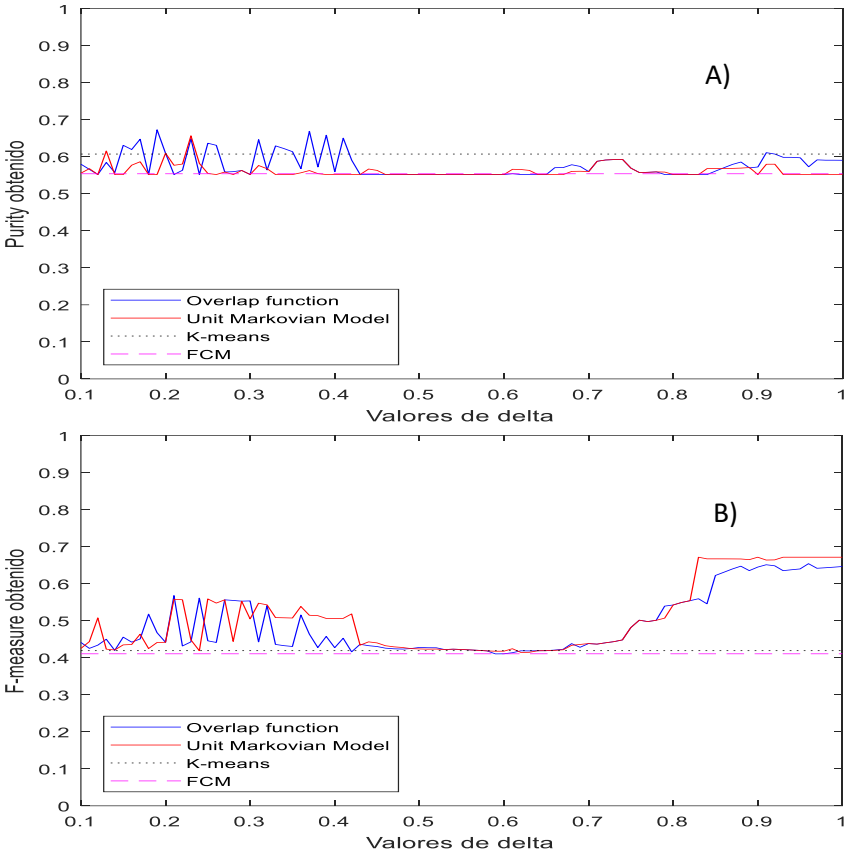


Fig. 13 Overlap 1 - p = 2 (A) Purity, B) F-measure)

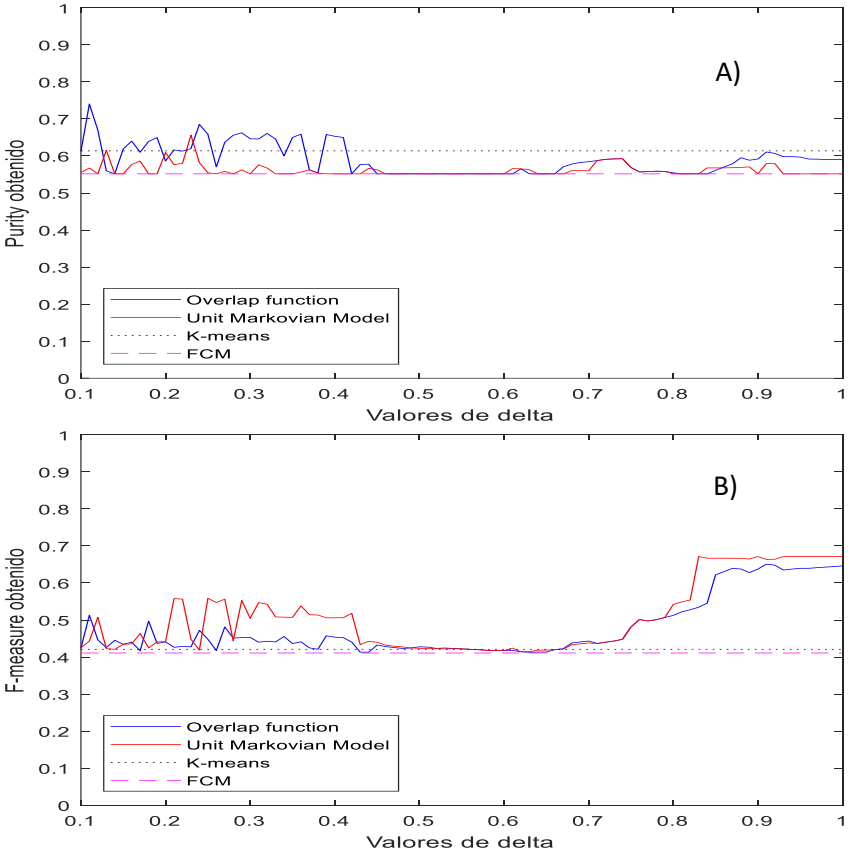


Fig. 14 Overlap 1 - p = 10 (A) Purity, B) F-measure)

Pruebas con datasets sintéticos

Tras las conclusiones iniciales inspiradas por los resultados anteriores se trató de profundizar en el comportamiento del algoritmo comprobando su efectividad al aplicarlo a algunos datasets artificiales de topología particular, a fin de comprobar en qué casos se obtenían mejores resultados.

Uno de los objetivos adicionales que se abordaron con estas pruebas fue determinar si existe una relación directa entre las mediciones obtenidas por evaluación interna y las obtenidas por evaluación externa. De ser así, sería posible emplear exclusivamente métodos de evaluación interna para precisar los mejores resultados ofrecidos por el algoritmo, sin necesidad de contar con una información a priori de la que a menudo se carece en casos reales.

A continuación se muestra una representación gráfica de los datasets empleados:

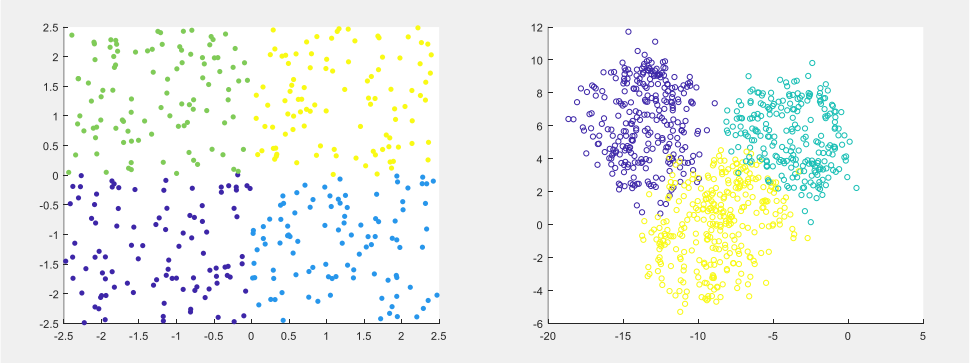


Fig. 15 Dataset "4 squares"

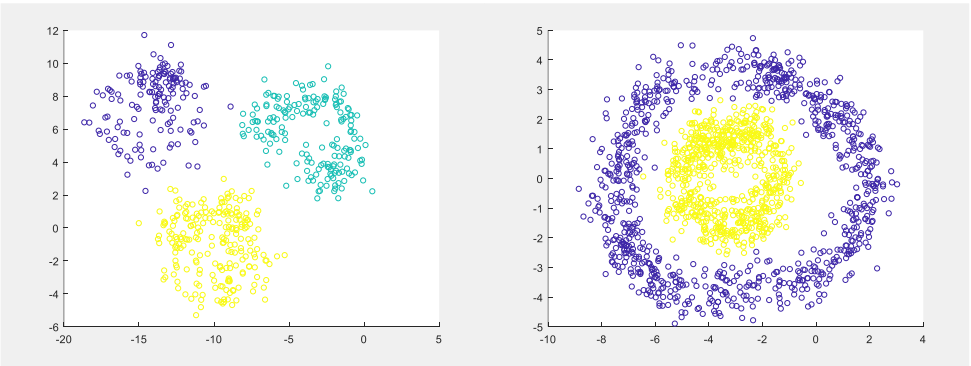


Fig. 16 Dataset "3 close circles"

Fig. 17 Dataset "3 circles"

Fig. 18 Dataset "2 concentric rings"

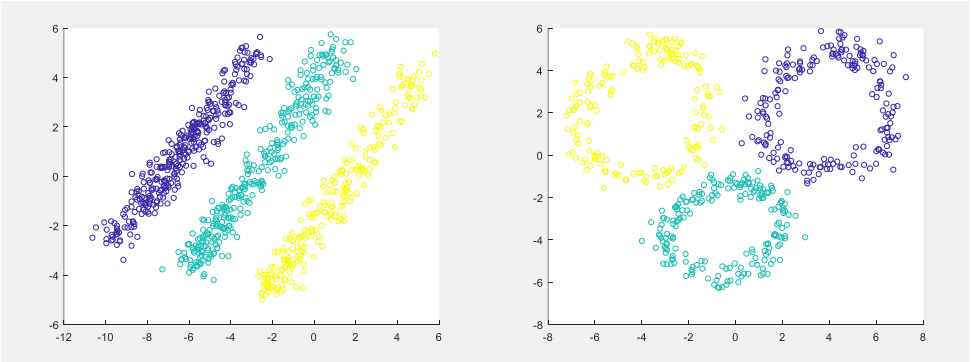


Fig. 19 Dataset "3 stripes"

Fig. 20 Dataset "3 rings"

Dataset "4 squares":

Todas las variantes del algoritmo devuelven peores resultados que los generados por el algoritmo FCM y el Kmeans. Esto puede deberse a la proximidad entre las fronteras de las clases, en las que probablemente se produzcan fusiones que lleven irremediablemente a una mala clasificación de algunos de dichos ejemplos. Pese a todo, los resultados son aceptables, con índices Rand o de pureza cercanos al 0,9.

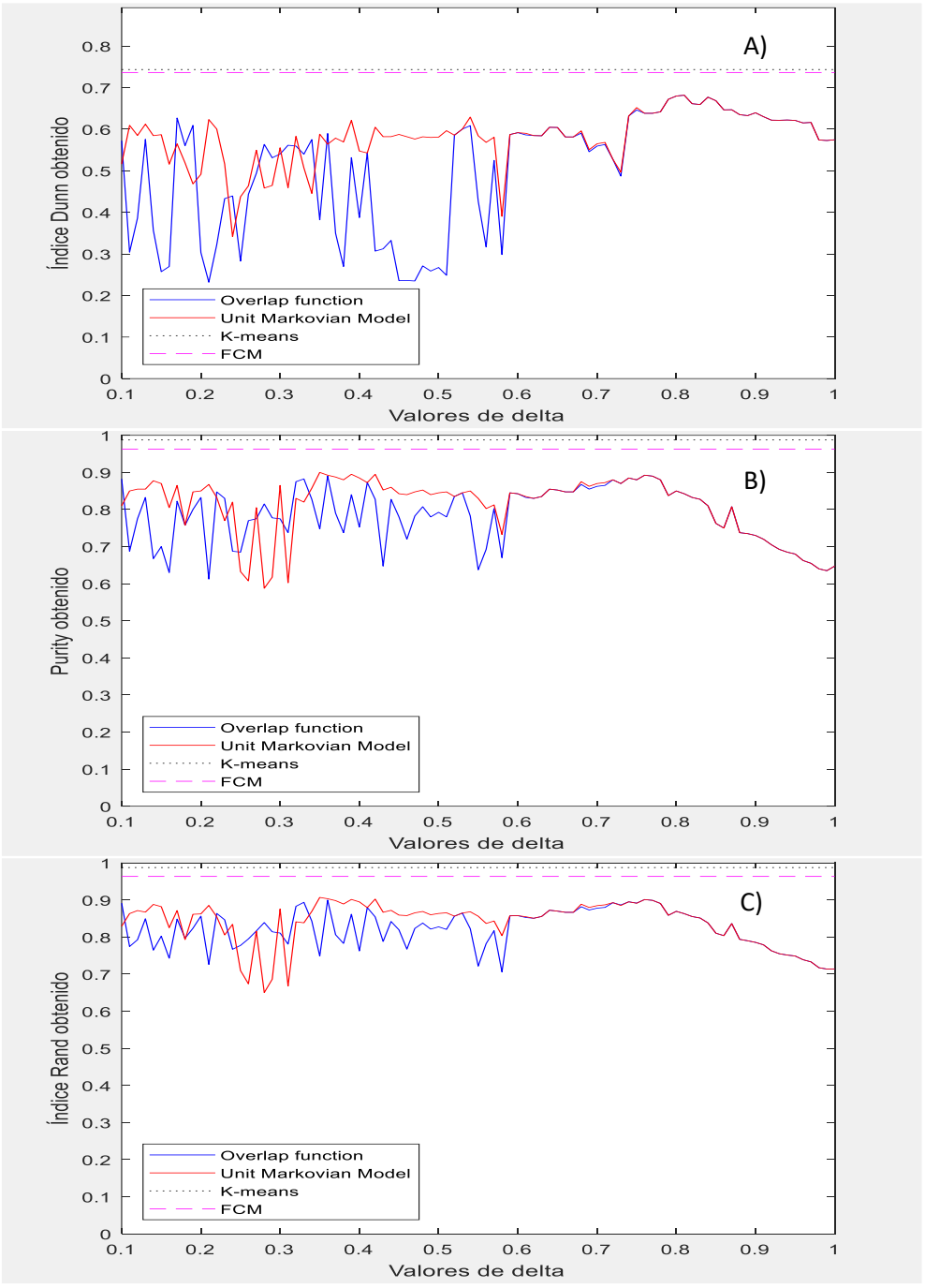


Fig. 21 Overlap 1 - p = 20 (A) Dunn-ndex, B) Purity, C) Rand-Index)

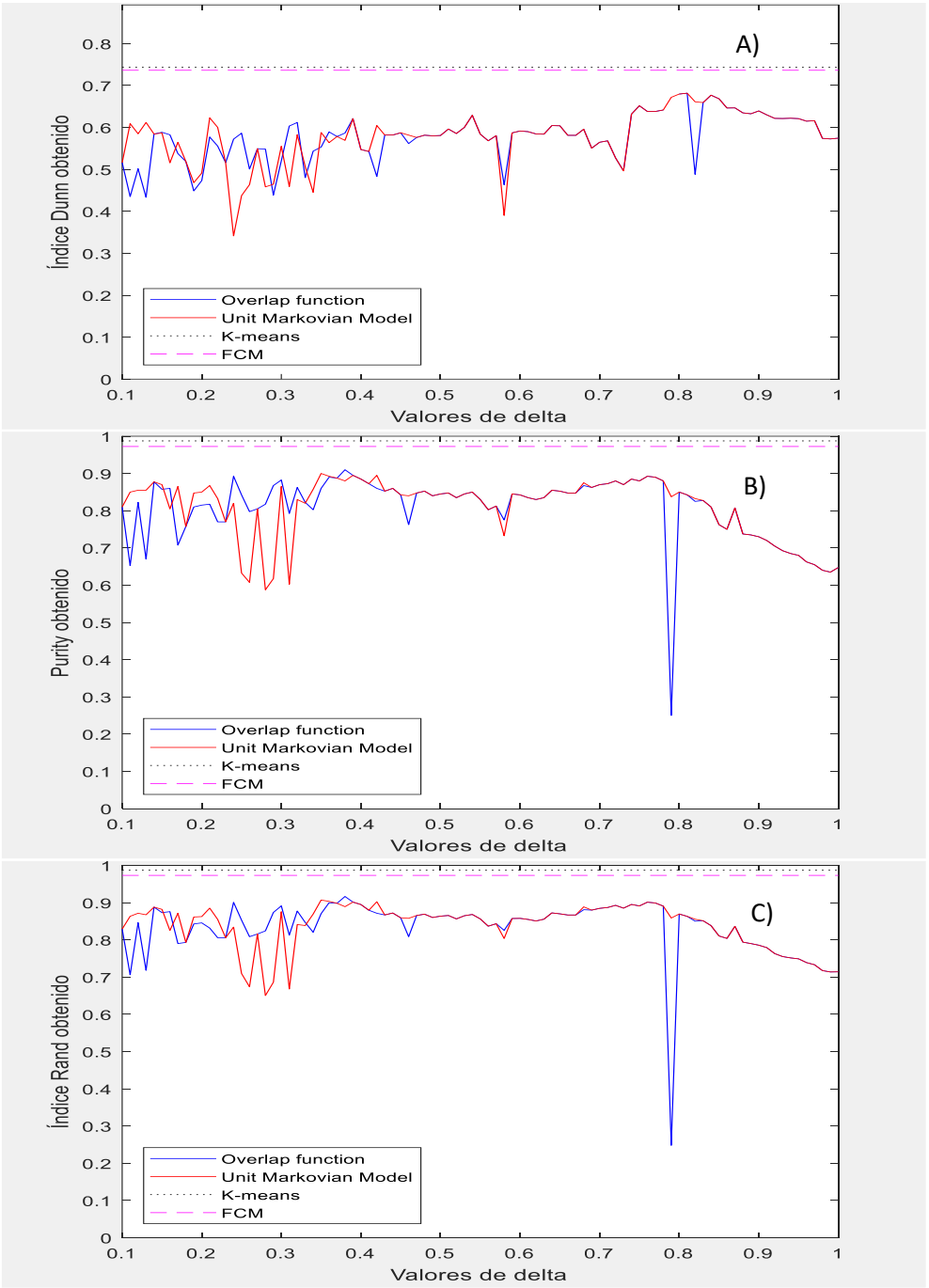


Fig. 22 Overlap 2 (A) Dunn-ndex, B) Purity, C) Rand-Index)

Dataset "3 close circles":

Una vez más nos encontramos con el mismo problema. A pesar de que el algoritmo obtiene resultados buenos, sigue perdiendo en media frente al algoritmo Kmeans y al FCM. No obstante, las variaciones utilizando overlaps obtienen clasificaciones cuya precisión supera a la obtenida por ambos algoritmos, para valores de delta concretos. Desgraciadamente, dichos valores no coinciden con los mayores índices Dunn (lo cual tiene sentido puesto que, en este caso, las mejores clasificaciones implican en algunas situaciones clasificar ejemplos en clústeres distintos a los que les correspondería por mera proximidad).

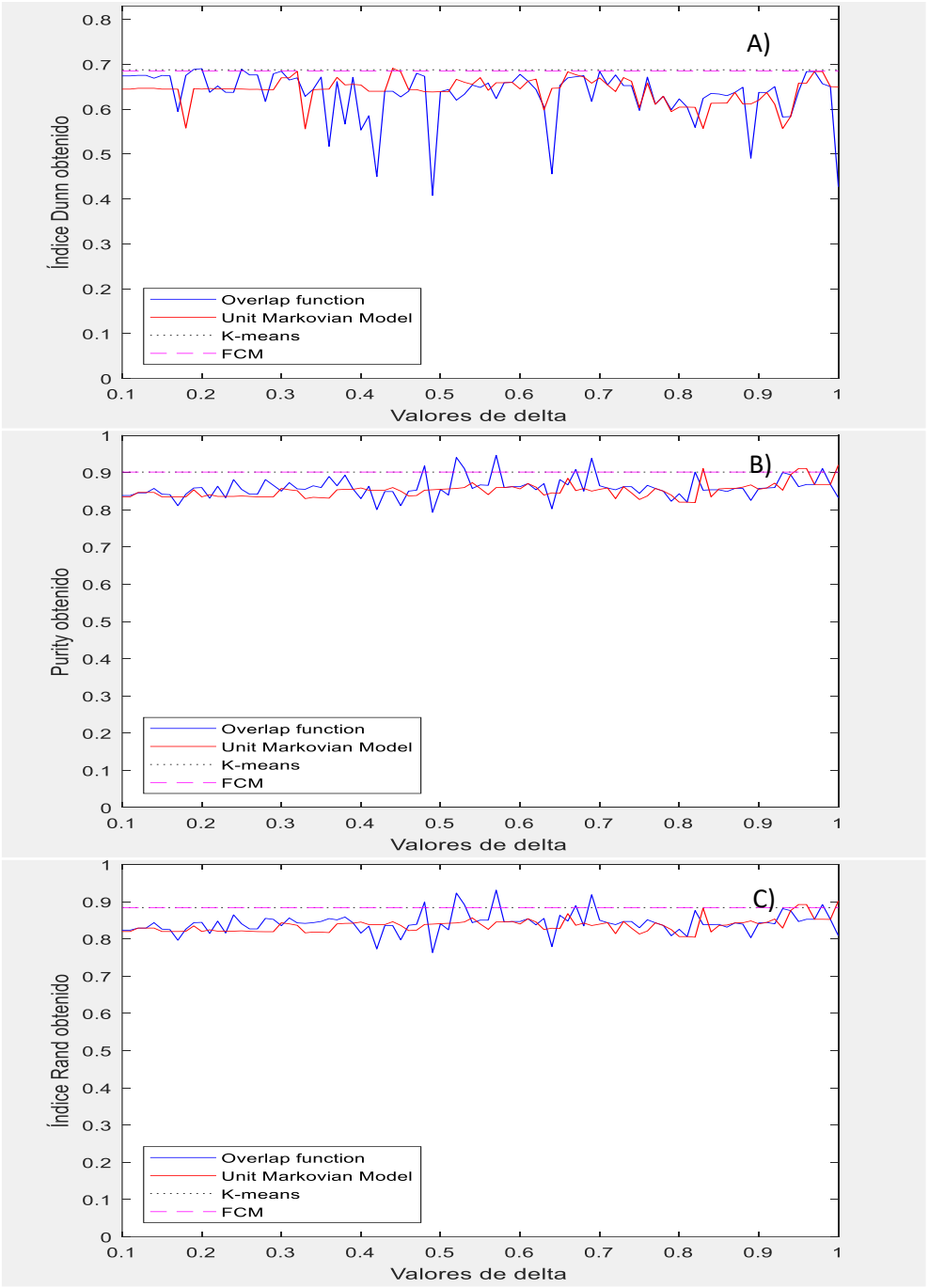


Fig. 23 Overlap 1 - p = 20 (A) Dunn-Index, B) Purity, C)Rand-Index)

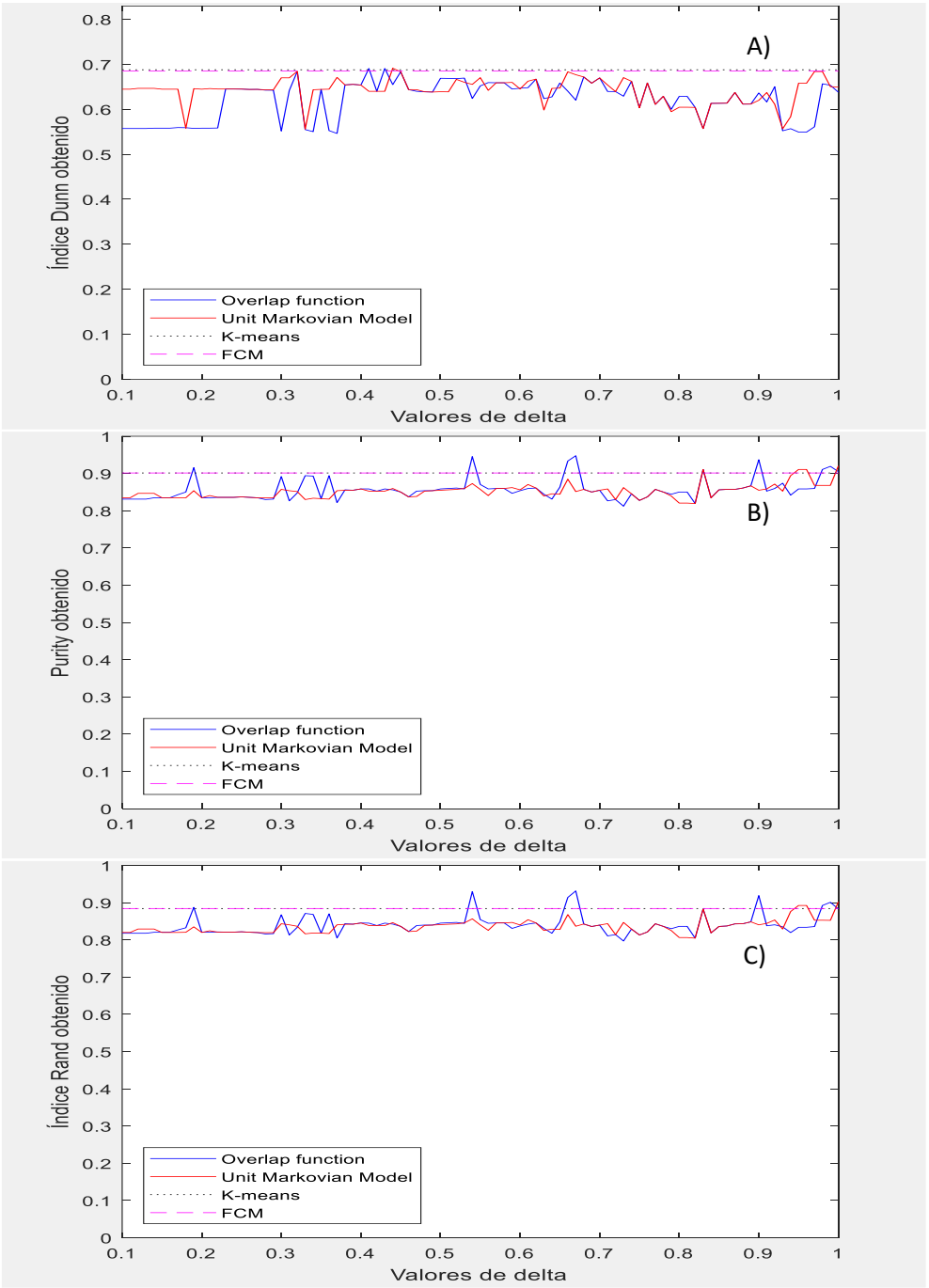


Fig. 24 Overlap 2 (A) Dunn-Index, B) Purity, C) Rand-index)

Dataset "3 circles":

En este caso, que resulta sencillo de clasificar, todos los métodos obtienen resultados casi perfectos, lo que refuerza la idea de que el algoritmo trabaja mejor con conjuntos compactos y separados, y particularmente con topología circular.

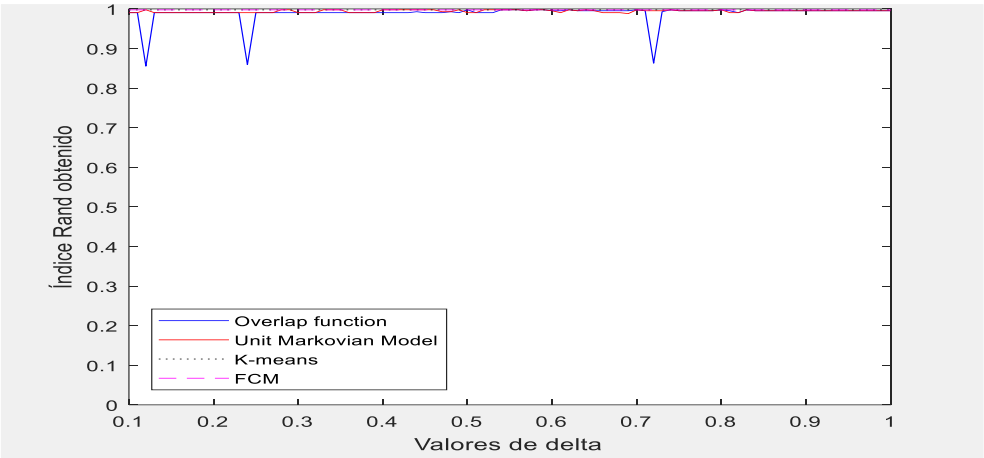


Fig. 25 Overlap 1 - p = 20 (Rand-Index)

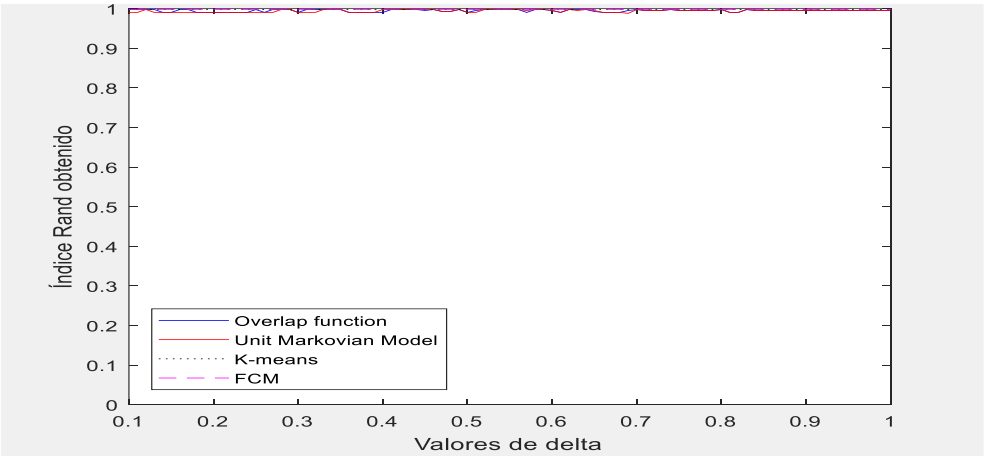


Fig. 26 Overlap 2 (Rand-Index)

Dataset "2 concentric rings":

Pese a que el algoritmo proporciona mejores resultados que el K-means y el FCM, en realidad las respuestas obtenidas distan de la clasificación ideal. Los clústeres que se obtienen preservan una forma circular en la que se incluyen ejemplos de ambas clases, lo que resulta lógico teniendo en cuenta el modo de trabajo del algoritmo unitario de Markov, que premia ante todo distancias cortas entre partículas a la hora de calcular sus velocidades.

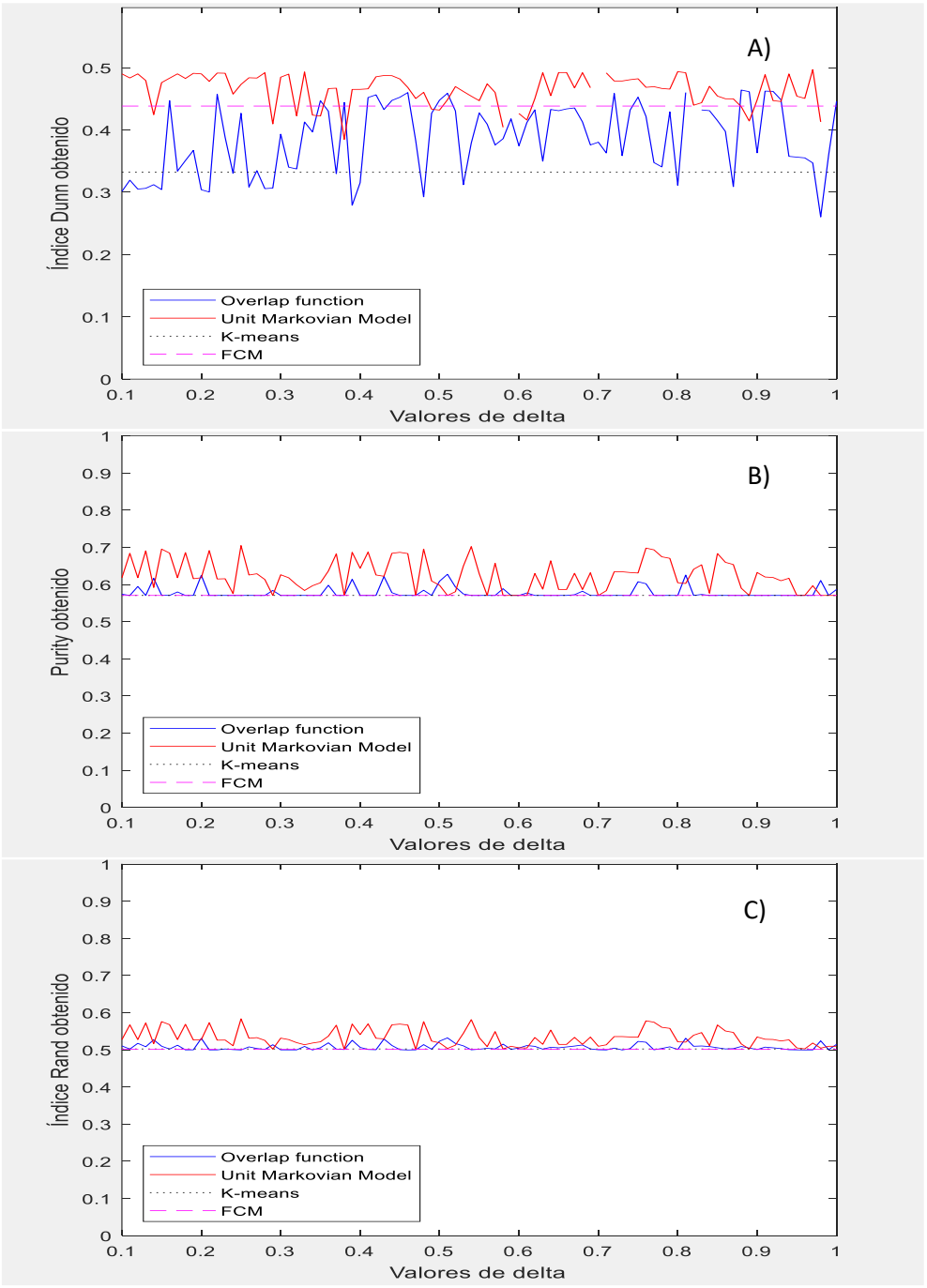


Fig. 27 Overlap 2 - p = 20 (A) Dunn-Index, B) Purity, C) Rand-Index)

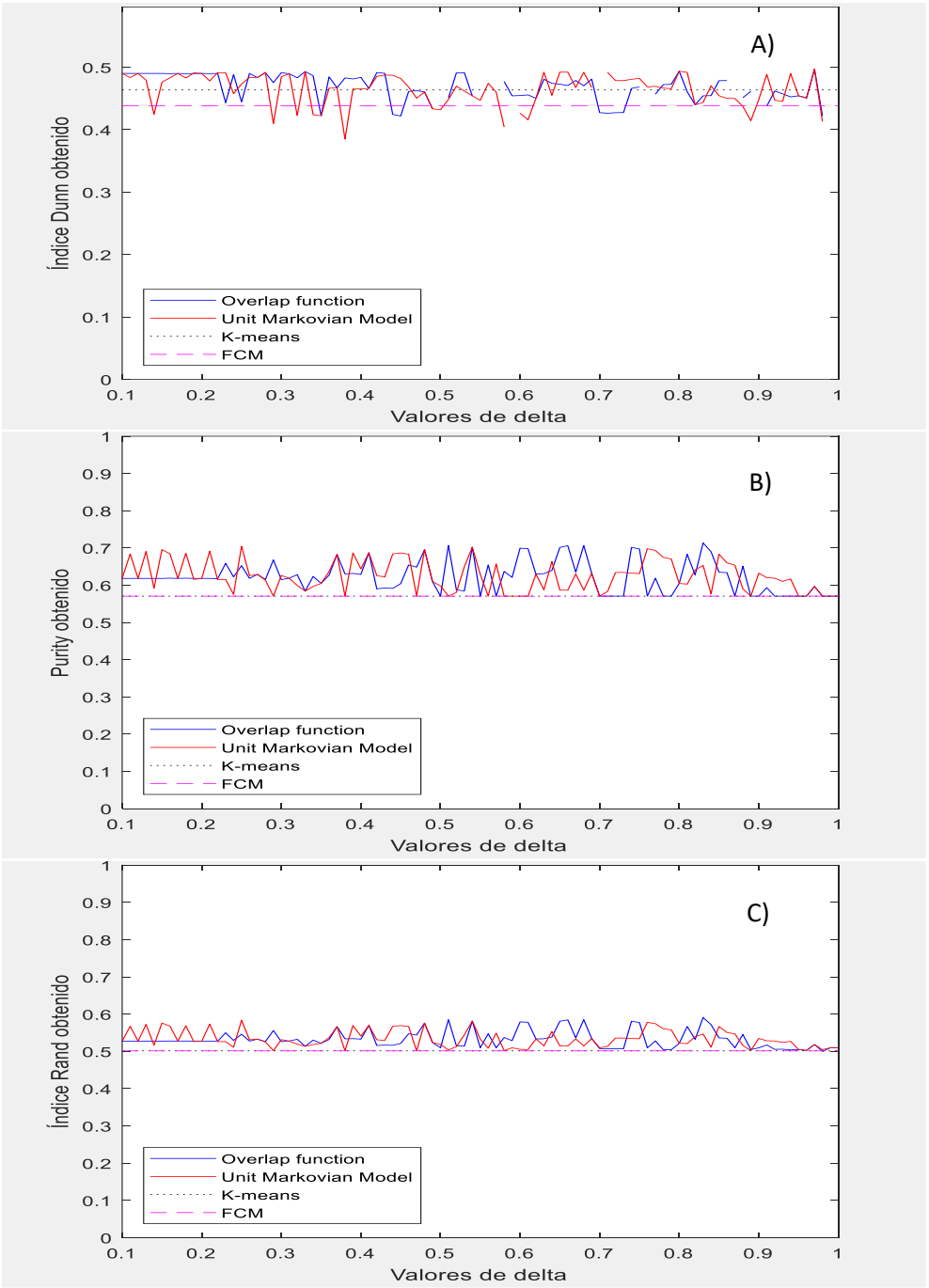


Fig. 28 Overlap 2 (A) Dunn-Index, B) Purity, C) Rand-Index)

Dataset "3 stripes":

En esta ocasión, a pesar de la gran variabilidad de los resultados, es para el overlap 1 para el que se obtienen los mejores, alcanzando valores de índice Rand en torno a 0,8 e incluso alcanzando el 0,9. El overlap 2 funciona de manera similar al modelo unitario de Markov, pero ninguna de estas variantes es capaz de superar a los algoritmos K-means o FCM.

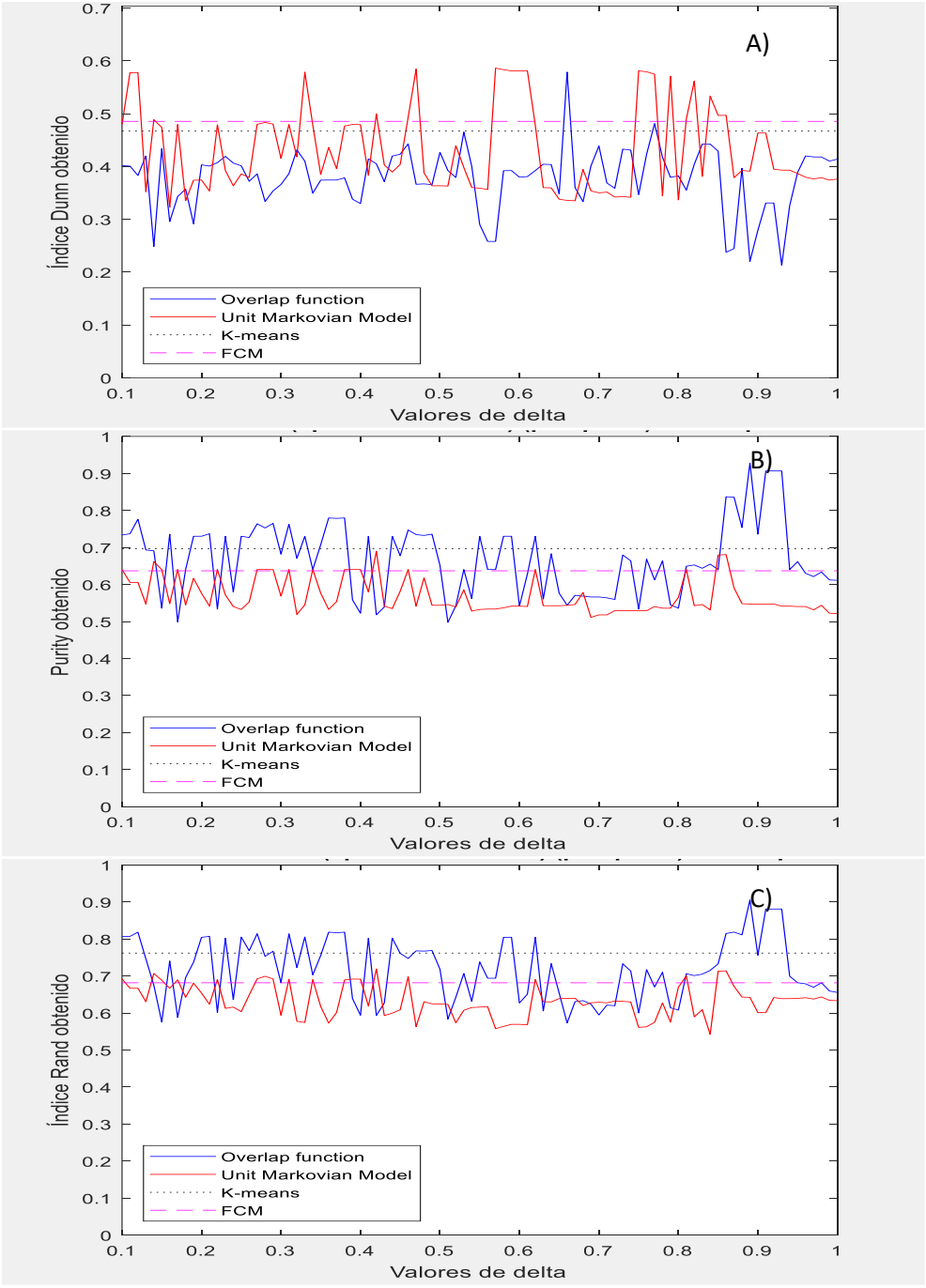


Fig. 29 Overlap 1 - p = 20 (A) Dunn-Index, B) Purity, C) Rand-Index)

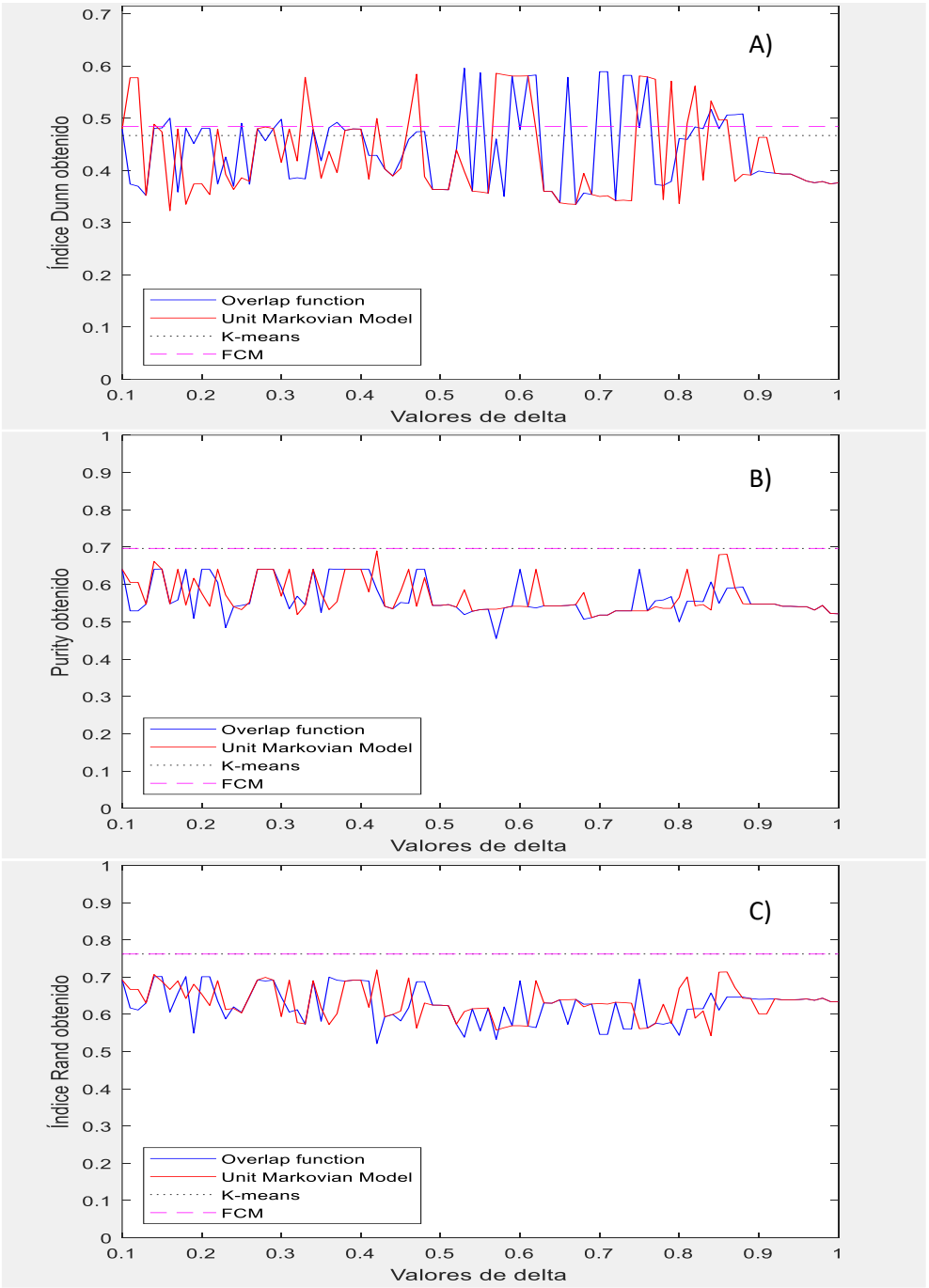


Fig. 30 Overlap 2 (A) Dunn-Index, B) Purity, C) Rand-Index)

Dataset "3 rings":

En general se obtienen mejores resultados empleando el K-means o el FCM, aunque el overlap 1 logra en algunos casos un clústering casi perfecto. Al estar tratando con anillos huecos muy próximos entre sí, es de esperar que las partículas de las fronteras entre clases, se atraigan generando clústeres erróneos que no coincidan con la clasificación a priori.

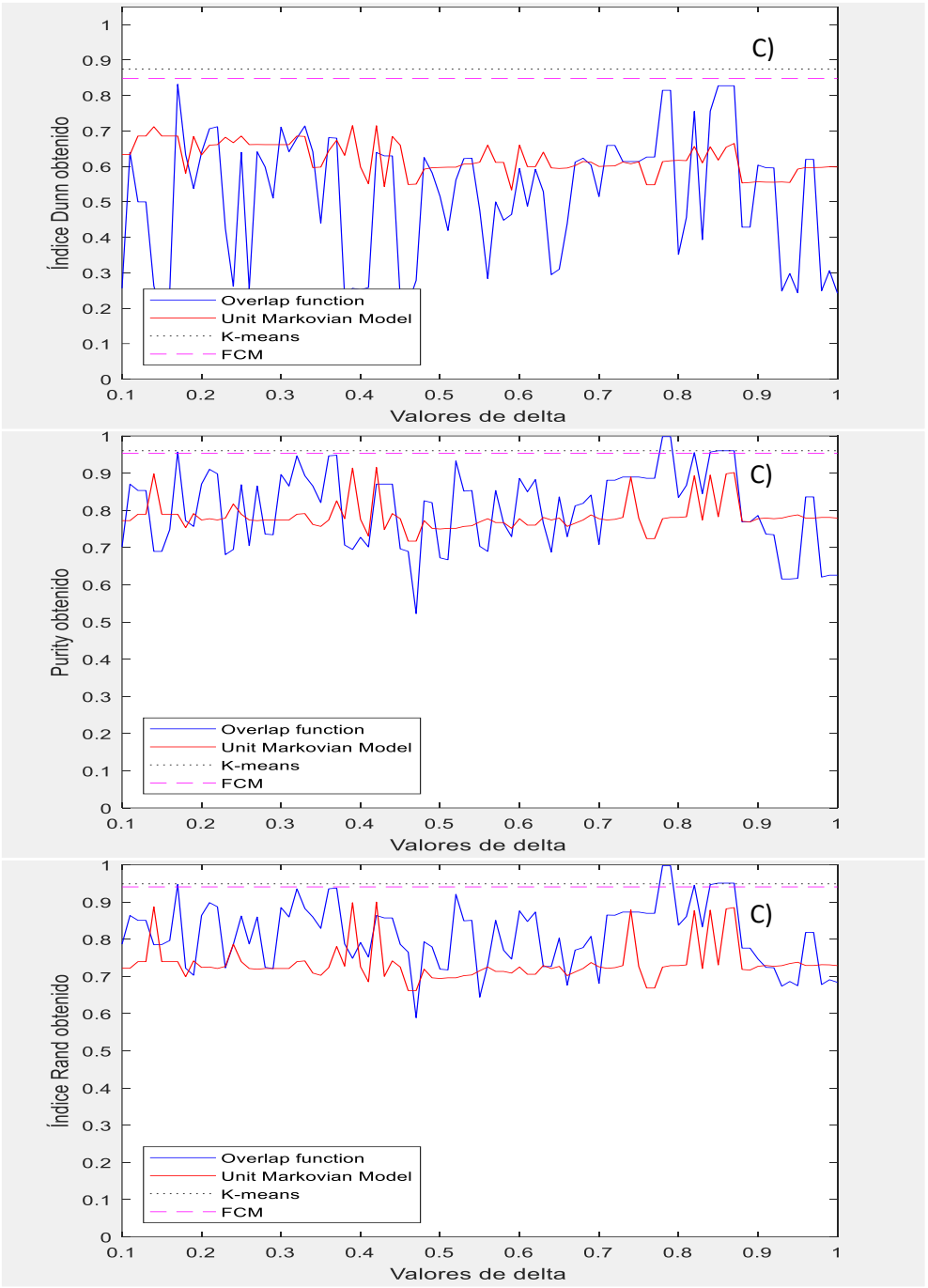


Fig. 31 Overlap 1 - p = 20 (A) Purity, B) Purity, C) Rand-Index)

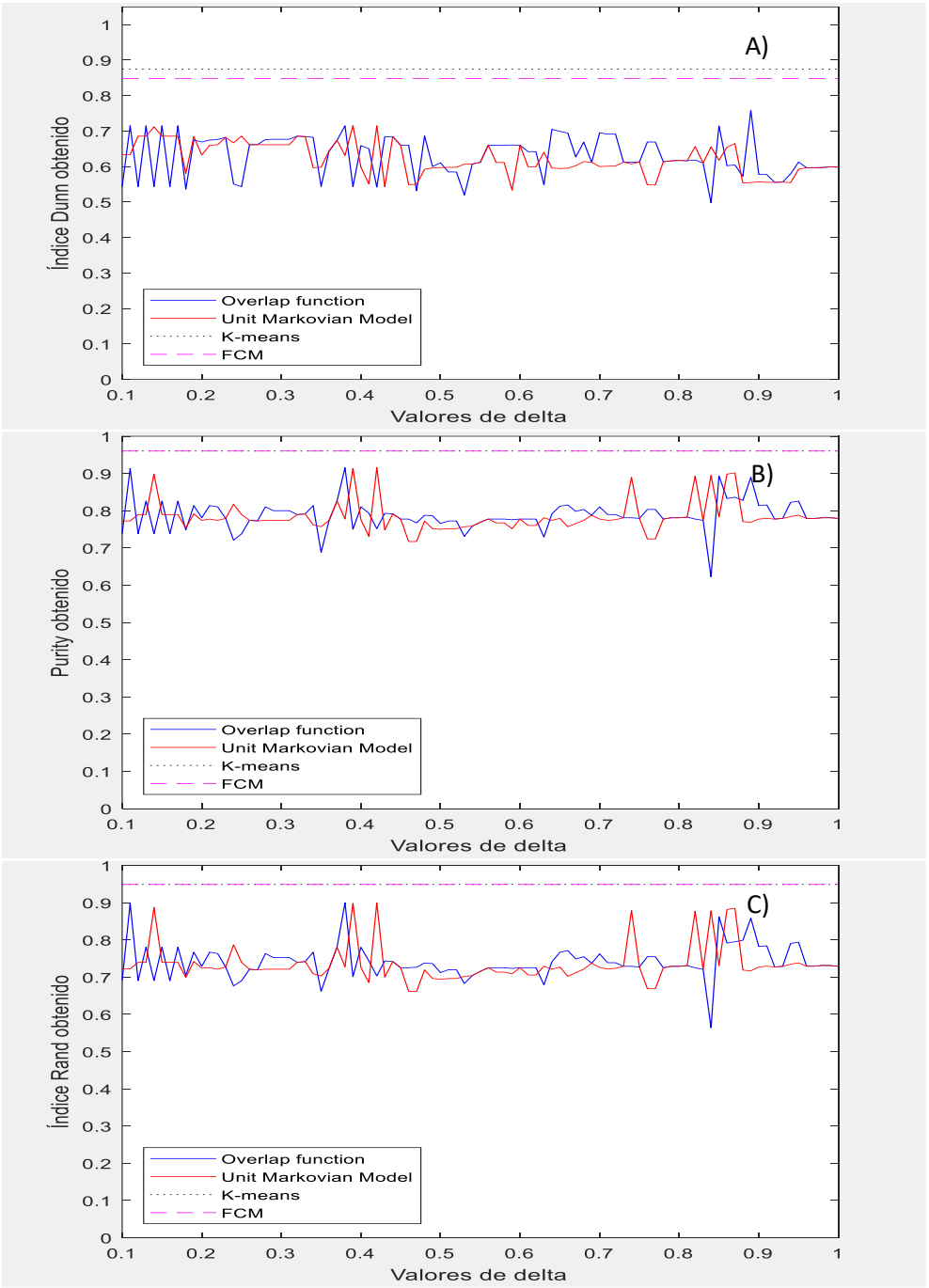


Fig. 32 Overlap 2 - (Rand-Index)

Resumen de los resultados:

En las siguientes tablas se sintetizan los valores obtenidos, tanto máximos como en media, para los datasets anteriores.

La diferencia entre los ganadores de la primera tabla y las otras dos deja claro que no existe una relación clara entre evaluación interna y evaluación externa, lo que supone un problema pues dificulta encontrar los mejores parámetros de p, δ y $\in$. Aun así, los valores de evaluación interna son decentes en los datasets de forma circular.

También se puede apreciar cómo el overlap 1 tiene la posibilidad de obtener los mejores valores, pero sin embargo es el que da lugar a una mayor varianza en los mismos. El overlap 2, por su parte, obtiene valores en media similares a los del modelo unitario, pero consigue los mejores valores en 3 ocasiones, aunque no es capaz de alcanzar los resultados del 1 para los datasets más particulares.

	OVERLAP 1		**OVERLAP 2**		**Unit Markovian Model**		**K-means**	**FCM**
Dataset	Mean	Max	Mean	Max	Mean	Max	Max	Max
4_squares	0,508	0,682	0,577	0,682	0,578	0,682	**0,743**	0,736
3_close_circles	0,633	0,690	0,621	**0,691**	0,641	**0,691**	0,688	0,685
3_circles	0,848	0,861	0,859	0,860	0,858	0,860	**0,932**	**0,932**
2_rings	0,386	0,464	0,468	**0,497**	0,464	**0,497**	0,464	0,438
3_stripes	0,378	0,579	0,442	**0,596**	0,432	0,586	0,467	0,485
3_separated_rings	0,519	0,831	0,631	0,758	0,621	0,715	**0,874**	0,848

Tabla 1 Comparativa de índices Dunn entre algoritmos

	OVERLAP 1		**OVERLAP 2**		**Unit Markovian Model**		**K-means**	**FCM**
Dataset	Mean	Max	Mean	Max	Mean	Max	Max	Max
4_squares	0,784	0,893	0,807	0,910	0,810	0,900	**0,988**	0,983
3_close_circles	0,859	0,946	0,856	**0,948**	0,852	0,920	0,901	0,901
3_circles	0,991	**0,998**	0,996	**0,998**	0,996	**0,998**	**0,998**	**0,998**
2_rings	0,577	0,628	0,622	**0,713**	0,624	0,705	0,570	0,570
3_stripes	0,670	**0,928**	0,564	0,642	0,569	0,690	0,697	0,690
3_separated_rings	0,806	**0,998**	0,788	0,916	0,784	0,916	0,960	0,960

Tabla 2 Comparativa de pureza entre algoritmos

	OVERLAP 1		**OVERLAP 2**		**Unit Markovian Model**		**K-means**	**FCM**
Dataset	Mean	Max	Mean	Max	Mean	Max	Max	Max
4_squares	0,822	0,901	0,834	0,917	0,838	0,907	**0,988**	0,983
3_close_circles	0,843	**0,931**	0,840	**0,931**	0,836	0,902	0,884	0,884
3_circles	0,989	**0,998**	0,995	**0,998**	0,994	**0,998**	**0,998**	**0,998**
2_rings	0,507	0,532	0,533	**0,591**	0,533	0,584	0,501	0,501
3_stripes	0,718	**0,905**	0,626	0,702	0,633	0,719	0,762	0,754
3_separated_rings	0,812	**0,998**	0,741	0,900	0,735	0,900	0,949	0,949

Tabla 3 Comparativa de Índices Rand entre algoritmos

5.3 Comportamiento del nuevo algoritmo:

A continuación se procederá a analizar más a fondo el comportamiento, tanto del algoritmo original, como de la versión empleando los overlaps 1 y 2, valiéndonos de representaciones visuales. Para ello, se hará uso de un dataset pequeño compuesto por 9 ejemplos, que se muestra a continuación:

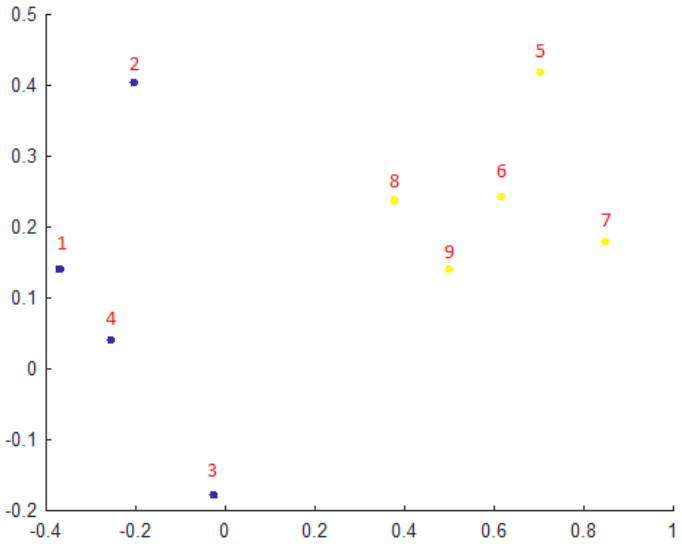


Fig. 33 Dataset "some points"

La figura [34] representa el movimiento de las partículas durante la ejecución del algoritmo. Cada uno de los puntos representa el estado de una partícula tras una iteración dada, siendo aquellos rodeados por una circunferencia roja los que indican las posiciones iniciales de cada una de ellas. Las fusiones entre partículas se representan con una marca roja y vienen acompañadas del número de la iteración en la cual han tenido lugar. Para este caso se emplearon los parámetros $\delta = 0.1, \epsilon = 0.2$, suficientemente buenos para lograr un clúster válido.

Como se puede apreciar, antes de alcanzar el bucle principal del algoritmo, se han producido ya 3 fusiones. Esto se debe a que las parejas de partículas (1, 4), (5, 6) y (8, 9) se encontraban a distancia inferior a ϵ y el algoritmo las ha combinado inicialmente, puesto que ya cumplían la condición para ser fusionadas. Estas partículas se desplazan, durante unas pocas iteraciones siguientes, en torno a la mitad de la distancia desplazada por el resto de partículas, debido a que su masa es el doble y, tal y como está definido el modelo unitario de Markov, la fuerza ejercida sobre una partícula es inversamente proporcional a su masa. Tras la iteración 3, momento en el cual se tienen ya los dos clústeres principales del algoritmo, las dos partículas resultantes invierten otras 4 iteraciones desplazándose la una hacia la otra, siendo el clúster de masa 4 el que recorre una mayor distancia por iteración.

La masa de las partículas también determina la posición en la que se genera la partícula resultante de una fusión, ya que esta se materializa en el centro de masas de ambas. Esto se puede apreciar comparando la posición de las colisiones ocurridas en la iteración 0, que genera la partícula resultante equidistantemente a cada una de las partículas iniciales de

masa 1, con las colisiones sucesivas, que se encuentran a menor distancia de la partícula con mayor masa.

El tiempo transcurrido entre iteraciones es también un parámetro importante de nuestro modelo, ya que será el que nos permita escoger el número de clústeres óptimo para cada dataset. A este respecto, la figura [35] representa a modo de dendrograma el tiempo transcurrido hasta cada una de las colisiones. Como se puede comprobar, las fusiones de las parejas de partículas (1, 4), (5, 6) y (8, 9) son instantáneas. Las siguientes colisiones se suceden de manera rápida, siendo la colisión final la que requiere en torno al 80% del tiempo del algoritmo para producirse. Por tanto, el número óptimo de clústeres para nuestro ejemplo será 2, tal y como era de esperar.

Es curioso observar, como la otra disposición de clústeres que más tiempo ha permanecido en el tiempo es la que daba lugar a 4 clústeres, siendo estos el compuesto por las partículas (5, 6, 7, 8, 9), el formado por (1, 4) y los dos restantes, compuestos cada uno por una única partícula. Esta distribución también tiene sentido, dado que las partículas 2 y 3 se encuentran relativamente alejadas del resto y podrían componer su propio clúster.

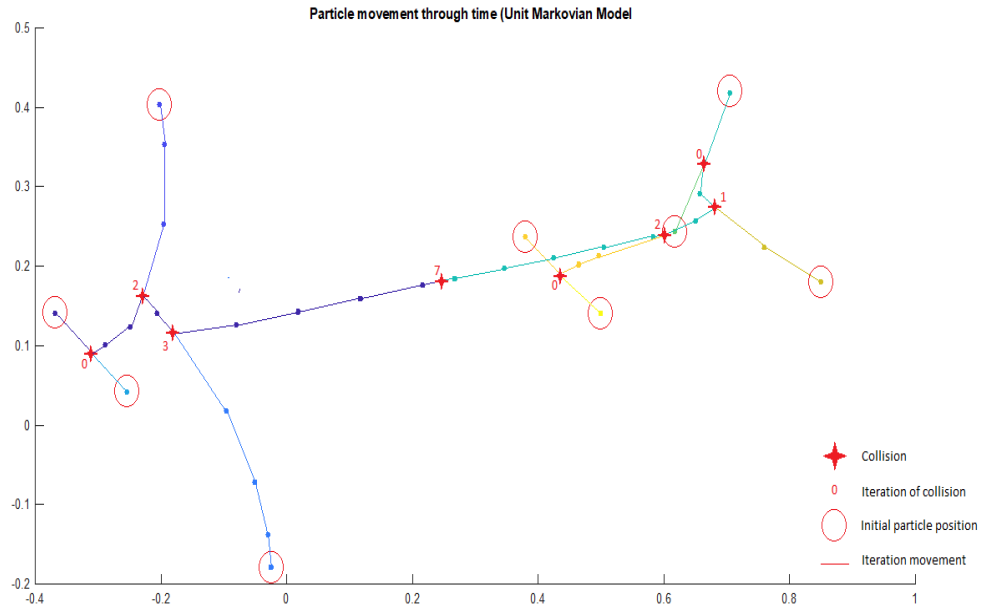


Fig. 34 Evolución del algoritmo (Modelo Unitario de Markov)

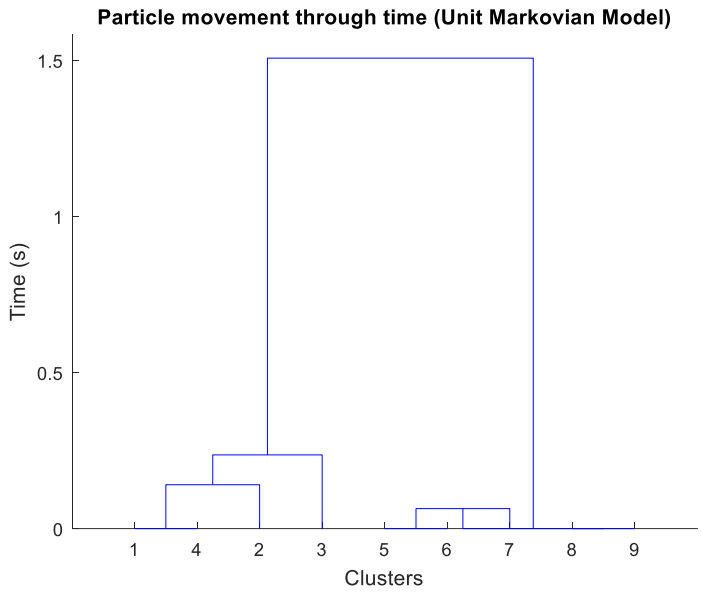


Fig. 35 Tiempo transcurrido entre colisiones (Modelo unitario de Markov)

Si pasamos ahora a observar el comportamiento del algoritmo cuando empleamos el overlap 1, vemos que, tal y como se observa en la figura [37], conforme el valor del parámetro p aumenta, el tiempo invertido entre colisiones lo hace también de forma exponencial. El motivo de este comportamiento se debe a que p se aplica como exponente de la masa de la partícula, de modo que a mayor p mayor penalización a las masas más grandes. Para p grandes, la fuerza aplicada sobre las partículas con mayor masa resulta ínfima en comparación al resto, y apenas se desplazan en cada iteración.

Esto también queda patente en la figura [36]. En especial, cuando $p = 20$, las partículas que colisionan se mantienen prácticamente estáticas hasta que todo el resto de partículas de masa inferior se han fusionado. Tras la iteración 5, momento en el cuál se ha formado el clúster compuesto por (5, 6, 7, 8, 9), dicho clúster permanece casi quieto hasta que, en la iteración 12, el clúster compuesto por (1, 2, 3, 4) se aproxima hasta colisionar con él. Como curiosidad, se puede observar que, cuando hay diferencia entre las masas, la distancia desplazada durante una colisión puede resultar significativamente mayor que el desplazamiento normal de la partícula, como sucede en este caso.

Pese a que esta vez, y dado a que el número de clústeres óptimo es 2, no existe problema, conviene tener en cuenta que al utilizar este overlap con p grande el método para calcular dicho número se puede ver afectado. La división alternativa que observábamos con el caso anterior y que daba lugar a cuatro clústeres es ahora inapreciable, por ejemplo.

El overlap 2, sin embargo, opera de manera similar al algoritmo clásico, dado que no penaliza tanto la diferencia entre masas. Su comportamiento se representa en la figura [38].

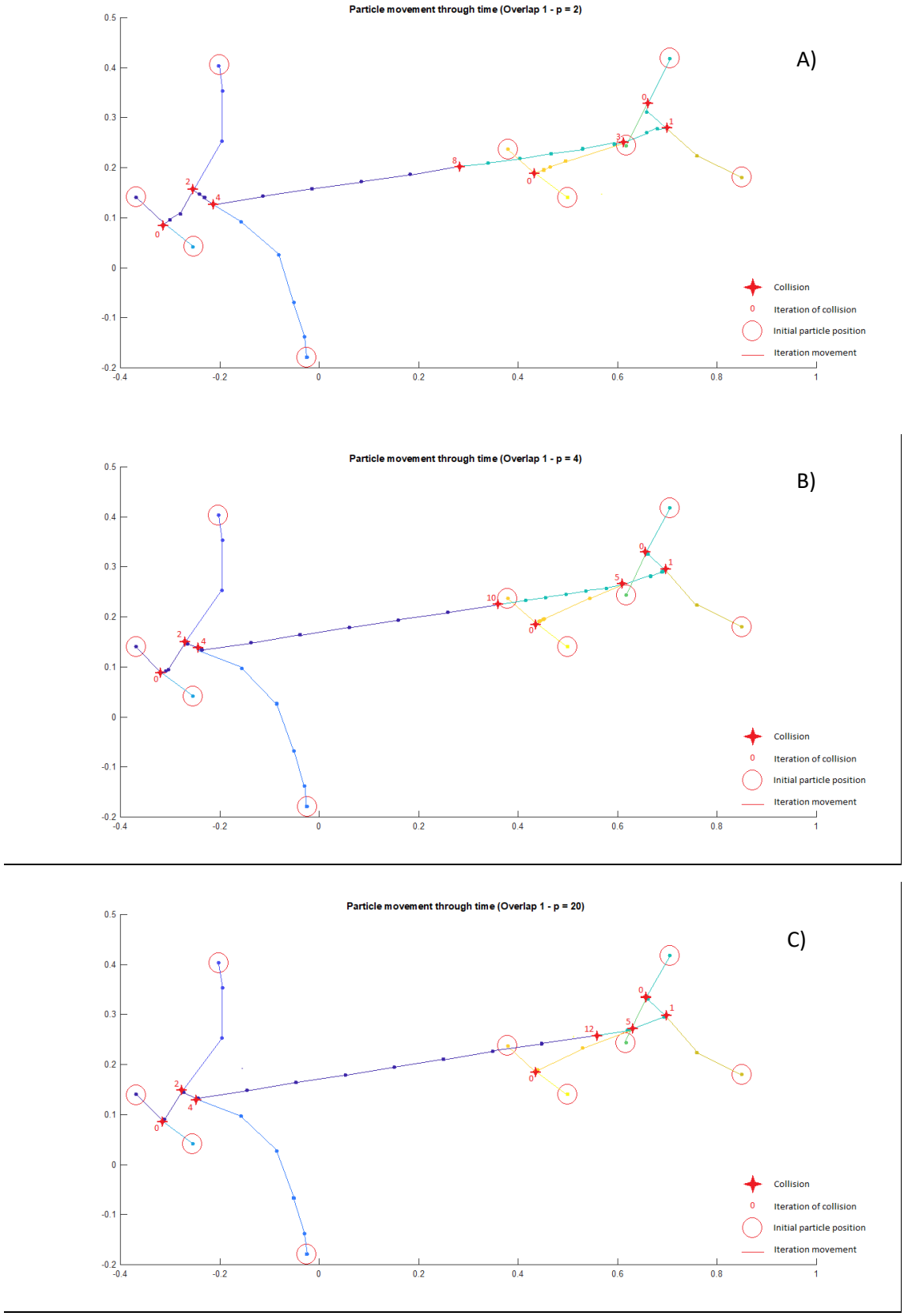


Fig. 36 Evolución del algoritmo – (Overlap 1 A) p = 2, B) p = 4, C) p = 20)

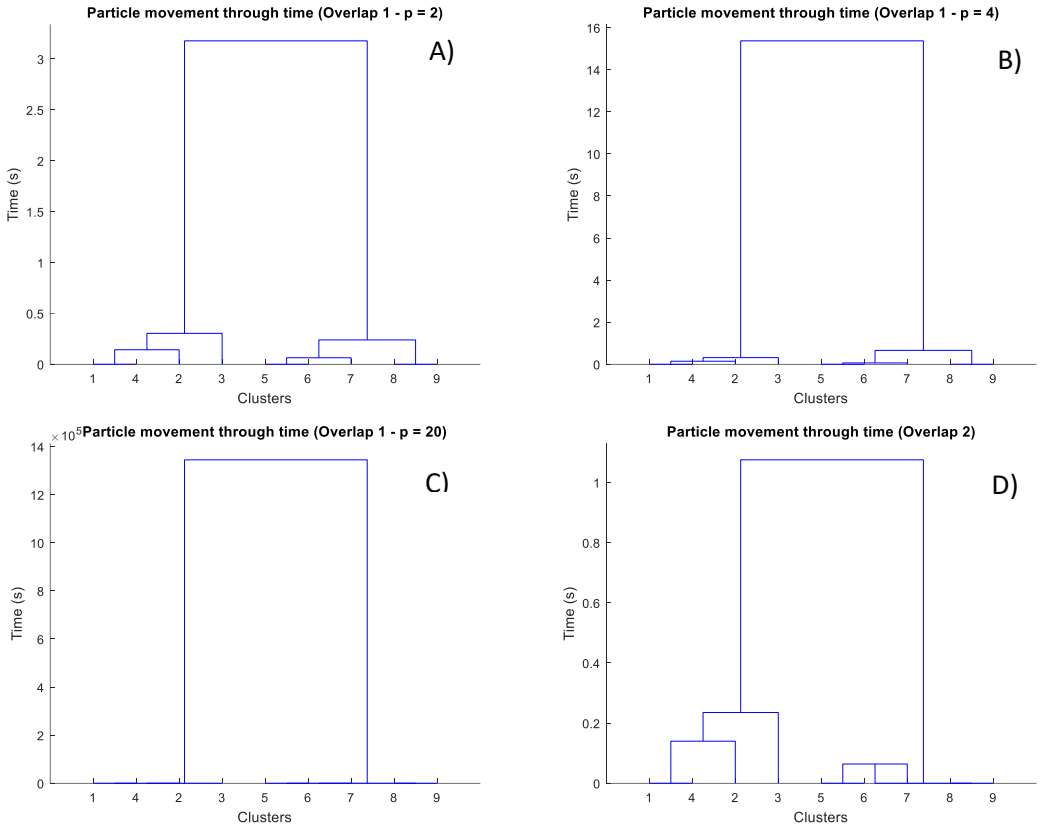


Fig. 37 Tiempo invertido entre colisiones (A) Overlap 1, p = 2, B) Overlap 1, p = 4, C) Overlap 1, p = 20, D) Overlap 2)

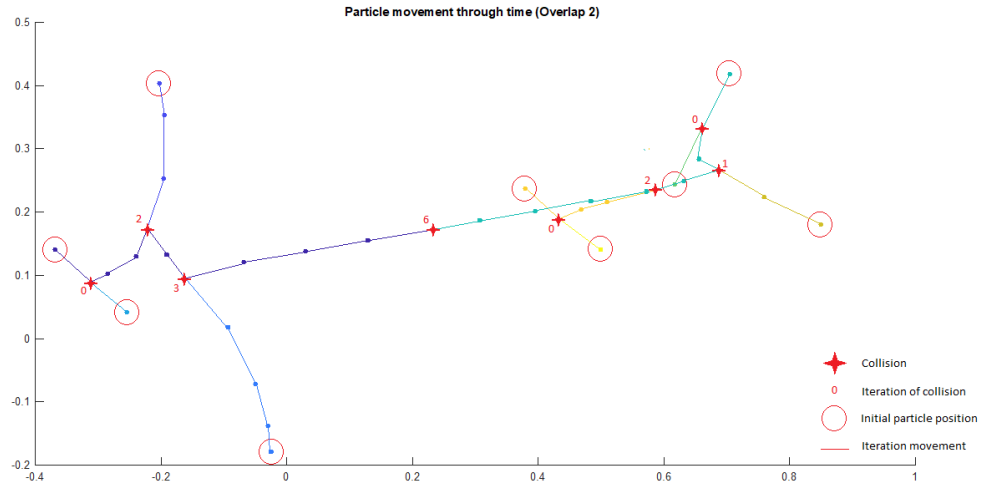


Fig. 38 Evolución del algoritmo – (Overlap 2)

6.- CONCLUSIONES

Los resultados parecen indicar que el algoritmo proporciona las mejores clasificaciones cuando trabaja con conjuntos de datos de topología circular, del mismo modo que lo hacen el algoritmo K-means y el FCM. Cuando las clases se encuentran claramente separadas, todos los algoritmos obtienen buenos resultados, que van empeorando en función de lo difusas que sean sus fronteras.

No obstante, un detalle importante que presenta el algoritmo gravitacional es que su modo de trabajo le permite obtener resultados aceptables en datasets de formas particulares para los cuales ni el K-means ni el FCM presentan buenos clústerings finales.

En cuanto a las variantes con overlaps, el overlap 2 parece obtener los resultados más estables para distintos valores de δ, alcanzando valores máximos que suelen mejorar o igualar los obtenidos por el modelo unitario de Markov. También es el que devuelve los mejores resultados para la evaluación interna en un mayor número de datasets. Resulta, por tanto, una buena alternativa al modelo clásico, que además permite mantener el método basado en la vida relativa de cada clasificación, a fin de encontrar la que más sentido tenga.

El overlap 1, por su parte, es el menos estable, aunque esta misma variabilidad es la que le permite alcanzar clústerings casi perfectos en algunos casos. Los resultados obtenidos resultan especialmente sorprendentes para el dataset “3 stripes”, para el cual se alcanzan índices de pureza y Rand del 0,928, valor al que no se acercan ninguno de los demás algoritmos probados. Por tanto, parece tener sentido emplear este overlap cuando se trabaja con clústeres de forma no circular. Esto no significa que no haya alternativas mejores que devieran priorizarse a la hora de tratar con este tipo de datasets, sino más bien que da una oportunidad al algoritmo gravitacional para tratar con ellos sin obtener resultados nefastos.

No obstante, el método tiene algunos problemas que conviene tener en cuenta:

En primer lugar, la varianza de los resultados supone un problema al tratar con casos en los que no se cuenta con una clasificación a priori, ya que encontrar los mejores valores para los parámetros es una tarea complicada que no tiene respuesta clara. Por lo general, en datasets circulares, a menor valor de δ mejor resultado, aunque los mejores resultados suelen encontrarse de manera puntual y no suelen cumplir dicha condición.

Por otro lado, el coste computacional del algoritmo resulta elevado, lo que dificulta su aplicación sobre datasets con muchos ejemplos o parámetros. También dificulta la posibilidad de repetir la ejecución del algoritmo con distintos parámetros a fin de encontrar la mejor combinación.

Con todo, las pruebas sugieren que el algoritmo gravitacional ofrece en general buenos resultados, que pueden llegar a mejorar los obtenidos tanto por el K-means como por el FCM. La elección de overlaps distintos le permite adaptarse a distintos datasets, y le da la posibilidad de lograr clasificaciones correctas en aquellos con topologías más peculiares.

BIBLIOGRAFÍA

[1] S. M. Aqil Burney y H. Tariq, «K-Means Cluster Analysis for Image Segmentation,» *International Journal of Computer Applications,* vol. 96, nº 4, 2014.

[2] K. Sarkar, «Sentence Clustering-based Summarization of Multiple Text Documents,» *International Journey of Computing Science and Comunication Technologies,* vol. 2, nº 1, 2009.

[3] R. Ducret, B. Lemarié y A. Roset, «Cluster Analysis and Spatial Modeling for Urban Freight. Identifying Homogeneous Urban Zones Based on Urban Form and Logistics Characteristics,» *Transportation Research Procedia,* vol. 12, pp. 301-313, 2016.

[4] D. Jiang, C. Tang y A. Zhang, «Cluster analysis for gene expression data: a survey,» *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, nº 11, pp. 1370-1386, 2004.

[5] W. E. Wright, «Gravitational Clustering,» *Pattern Recognition,* vol. 9, pp. 151-166, 1977.

[6] D. H. Wolpert y W. G. Macready, «No free lunch theorems for optimization,» *IEEE Transactions on Evolutionary Computation,* vol. 1, nº 1, pp. 67-82, 1997.

[7] A. Williams, «Its neuronal,» 11 Septiembre 2015. [En línea]. Available: http://alexhwilliams.info/itsneuronalblog/2015/09/11/clustering1/. [Último acceso: 3 Junio 2018].

[8] Scikit-learn, «2.3. Clustering,» Scikit-learn, [En línea]. Available: http://scikit-learn.org/stable/modules/clustering.html. [Último acceso: 3 Junio 2018].

[9] Contribuyentes de Wikipedia, «Cluster analysis,» Wikipedia, The Free Encyclopedia, 10 Mayo 2018. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=840515260. [Último acceso: 3 Junio 2018].

[10] K. Wong, «A Short Survey on Data Clustering Algorithms,» 2015. [En línea]. Available: doi: 10.1109/ISCMI.2015.10..

[11] Suprihatin, T. R. Y. Iwan, I. Nursyiva, P. Tuti, Haviluddin y P. W. Aji, «A Performance of Modified Fuzzy C-Means (FCM) and Chicken Swarm Optimization (CSO),» de *3rd International Conference on Science in Information Technology (ICSITech)*, Bandung, Indonesia, 2017.

[12] S. Kundu, «Gravitational clustering: a new approach based on the spatial distribution of the points,» *Pattern Recognition,* vol. 32, nº 7, pp. 1149-1160, 1999.

[13] K. Blekas y I. E. Lagaris, «Newtonian clustering: An approach based on molecular dynamics,» *Pattern Recognition,* vol. 40, nº 6, pp. 1734-1744, 2007.

[14] L. Junlin y F. Hongguang, «Molecular dynamics-like data clustering approach,» *Pattern Recognition,* vol. 44, nº 8, pp. 1721-1737, 2011.

[15] D. Arthur y S. Vassilvitskii, «K-Means++: The Advantages of Careful Seeding,» de *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, 2007.

[16] B. J. C., R. Ehrlich y W. Full, «FCM: The Fuzzy c-Means Clustering Algorithm,» *Computer & Geosciences,* vol. 10, nº 2-3, pp. 191-203, 1984.

[17] D. M. Manning, P. Raghavan y H. Schütze, «Evaluation of clustering,» Stanford University, 7 Abril 2009. [En línea]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html. [Último acceso: 3 Junio 2018].

[18] H. Bustince, J. Fernandez, R. Mesiar, J. Montero y R. Orduna, «Overlap Functions,» de *Nonlinear Analysis, 72*, 2010, pp. 1488-1499.

[19] D. Dua, K. Taniskidou y Efi, «UCI Machine Learning Repository,» University of Carolina, Irvine, School of Information and Computer Sciences, 2017. [En línea]. Available: http://archive.ics.uci.edu/ml. [Último acceso: 3 Junio 2018].

[20] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez y F. Herrera, «KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework,» *Journal of Multiple-Valued Logic and Soft Computing,* vol. 17, nº 2-3, pp. 255-287, 2011.

ANEXO: Propuesta de artículo derivado a partir de los resultados

A Generalization of the Gravitational clustering algorithm Using Overlap Functions

J. Armentia[1], I. Rodriguez[1], M. Minarov [1], J. Ruiz-Aranguren[1],

J.Fernandez[1], H. Bustince[1]

1. Introduction

Given a set of instances, clustering consist of determining, in an unsupervised way, one or several classes (clusters) into which they can be partitioned [3]. Many different methods have been proposed in the literature to deal with this problem, due to its high applicability [8]

In 1977, Wright [9] proposed a clustering algorithm based on the use of Newtons gravitational law. In his proposal, instances were assumed to be points in an Euclidean space of a given dimension, that is, real vectors of n components for some n>0. Then, a mass of one is assigned to each of the particles. In this way, acceleration can be calculated by means of Newton's law

$$g(m_i, m_j) = \frac{1}{2} G \sum_{j \neq i} \frac{m_i(t) m_j(t)}{m_i(t)} \frac{s_j(t) - s_i(t)}{\left| s_j(t) - s_i(t) \right|^3}$$

can be applied, where $s_i(t)$ and $s_j(t)$ are the vectors representing instances i and j, respectively, at some moment t .

The expression in the numerator can be seen as a particular case of overlap functions. Overlap functions were initially introduced in the fuzzy setting in order to deal with the problem of determining to which of two overlapping sets a given input belongs more [1]. But, since its inception, overlap functions have been successfully applied in fields such as image processing [4], classification [2, 6] or optimization [5].

In particular, In [5] a very competitive generalization of the GSA, an optimization algorithm which makes also use of the gravitational force, see [7], was proposed, replacing the product in the expression of the force by overlap functions (and some other types of functions).

Email addresses: bustince@unavarra.es (J. Armentia), bustince@unavarra.es (I. Rodriguez), maria.minarova@stuba.sk (M. Minárová), jruizaranguren@gmail.com (J. Ruiz-Aranguren), bustince@unavarra.es (J. Fernandez), bustince@unavarra.es (H. Bustince)

The objective of this work is to generalize the gravitational clustering algorithm considering general overlap functions instead of the product in the expression of the force Observe that such a generalization will recover as a particular case (taking as overlap function the product) Wright's original algorithm. Furthermore, and in order to show the validity of our proposal: 1.-We will prove theoretically under which conditions our generalized version using general overlap function is convergent, and 2.-We will experimentally compare our generalized version of the algorithm, which includes as a particular case the original one, both to the original version and to some of the most widely used in the literature to show that in some cases our algorithm overcomes all the other ones.

The structure of the paper is the following. We start with some preliminary notions and concepts that are needed for the rest of the paper. In Section 3 we discuss of our proposal of a generalized version of the gravitational clustering algorithm. In Section 4 we provide an experimental analysis to show that our proposal overcomes in some cases both the original version of the algorithm and some of the most widely used clustering methods in the literature. We finish with some conclusions and references.

2. Preliminars

2.1. Aggregation functions

Definition 2.1. An n-dimensional function $M: [0,1]^n \to [0,1]$ is called aggregation function if it fulfils following conditions:

- M is non increasing in each of its variable;
- $M(0, \dots, 0) = 0$ and $M(1, \dots, 1) = 1$

Examples: Averages, product, minimum, maximum...

Definition 2.2. Let $M: [0,1]^n \to [0,1]$ be an n-dimensional aggregation function

(i) $a \in [0,1]$ is an annihilator of M if $M(x_1, \dots, x_n)$ whenever $a \in \{x_1, \dots, x_n\}$.

(ii) If M does not have annihilator, then M is called strictly increasing as a real function of n variables on its domain $[0,1]^n$ if it is increasing in each of n variables. If M has an annihilator a, the strictly increase is done on $([0,1]\backslash\{a\})^n$.

(iii) M has zero divisors if there exist $x_1, \dots, x_n \in [0,1]$ such that $M(x_1, \dots, x_n) = 0$.

(iv) M is idempotent if $M(x, \dots, x) = x$ for all $x \in [0,1]$.

Next we focus on the bivariate aggregation functions. Following proprieties will be of our interest.

Definition 2.3. Let $M: [0,1]^n \to [0,1]$ be an aggregation function of two variables.

(i). M is symmetric if $M(x,y) = M(y,x)$ for any $x, y \in [0,1]$.
(ii). M is associative if $M(M(x,y),z) = M(x,M(y,z))$ for any $x, y, z \in [0,1]$

Associativity makes it possible to extend bidimensional aggregation functions to bigger dimensions in a "reasonable" manner. In particular, it's one of the fundamental properties needed in the construction of triangular norms (t-norms).

Definition 2.4. A t-norm (triangular norm) is a bivariate aggregation function $T: [0,1]^n \to [0,1]$ which is symmetric, associative and $T(1,x) = x$ for all $x \in [0,1]$.

T-norms allow to model intersections among fuzzy distributions. However, if we're working with two single elements, associativity isn't a needed property, in principle. This idea of being able to represent the intersection without necessarily requiring associativity brings us to the concept of overlapping function, which we'll study now.

2.2. Overlap functions

In this section we bring the results of [1].

Definition 2.5. A function $O: [0,1]^2 \to [0,1]$ is an overlap function if

(O1). O is symmetric;

(O2). $O(x,y) = 0$ if and only if $xy = 0$;

(O3). $O(x,y) = 1$ if and only if $xy = 1$;

(O4). O is non-decreasing;

(O5). O is continuous.

Following two functions are examples of overlap functions.

- $O(x,y) = \min(x,y)$
- $O(x,y) = xy$

These functions are t-norms in particular. Actually, each continuous t-norm without zero divisors ($T(x,y) = 0$ is hold only if $xy = 0$) is an overlap function. However, there exist overlap functions that are not t-norms, for example:

$$O(x,y) = \min(xy^k, x^k y) \text{ with } k > 0$$

which is not a t-norm if $k \neq 1$,

$$O(x,y) = (xy)^p \text{ with } p > 0$$

which is not a t-norm if $p \neq 1$, or

$$O(x,y) = \sin\frac{\pi}{2}(xy)^p \text{ with } p > 0$$

which is not a t-norm at all.

We can now summarize the relation between overlap functions and t-norms.

Theorem 2.6. *Let O be an associative overlap function. Then O is a t-norm.*

In constructing an overlap function, it is possible to proceed in accordance to the following theorem.

Theorem 2.7. *Function* $O\colon [0,1]^n \to [0,1]$ *is an overlap function if and only if*

$$O(x,y) = \frac{f(x,y)}{f(x,y) + h(x,y)}$$

with $f, h\colon [0,1]^2 \to [0,1]$ *such that*

1) f *and* h *are symmetric;*
2) f *is non-decreasing and* h *is non-increasing;*
3) $f(x,y) = 0$ *if and only if* $xy = 0$*;*
4) $h(x,y) = 0$ *if and only if* $xy = 1$*;*
5) f *and* h *are continuous;*

For example, having $f(x,y) = \sqrt{xy}$ and $h(x,y) = \max(1-x, 1-y)$, we obtain an overlap function

$$O(x,y) = \frac{\sqrt{xy}}{\sqrt{xy} + \max(1-x, 1-y)}$$

2.3. Algorithm of Gravitational Clustering

Algorithm of Gravitational Clustering employs the Newton gravitational law within the process of clustering. The scheme is as follows.

We suppose that we have n particles $p_1, \dots, p_n$, with their positions $s_1, \dots, s_n \in \mathbb{R}^n$.

1. Initially we assign a mass 1 to each particle p_i.
2. We fix real positive parameters ϵ and δ
   - We utilize δ for determining the actual time step of the actual dt. Indeed, in the lapsing time $[t, t+dt]$, the most rapid particle moves by δ.
   - If in a moment two particles find themselves in a distance less than ϵ, we unify them in one with mass equal to sum of masses of both of them and position done by their center of gravity.
3. Initial time $t = 0$ is set.
4. We repeat following steps (i)-(iv) until a single particle remains.
   (i). In each time interval $[t, t+dt]$, for each particle i we compute its movement influencing function:

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} \frac{m_i(t) m_j(t)}{m_i(t)} \frac{s_j(t) - s_i(t)}{\left|s_j(t) - s_i(t)\right|^3} dt^2 \qquad (1)$$

   Where G is a positive constant.

   (ii). For each particle i, its new position is:

$$s_i(t+dt) = s_i(t) + g(i,t,dt)$$

   (iii). We raise t to $t + dt$.
   (iv). If two particles i and j are in a distance less than ϵ, their unification is done as explained above.

Finally we have just one particle. The duration of the entire process is denoted by T and it evolved as follows:

- From $t_n = 0$ to t_{n-1} there are n (all) particles remaining.
- From t_{n-1} to t_{n-2} there are $n-1$ particles remaining.
- …
- From t_3 to t_2 there are 3 particles remaining.
- From t_2 to $t_1 = T$ there are 2 particles remaining.

The relative life of the configuration with k clusters can be computed as

$$R_k = \frac{t_k - t_{k-1}}{T}$$

This model described above can be generalized by using a more general expression for the particle movement governing function instead of (1).

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} \frac{m_i(t)^p m_j(t)^q}{m_i(t)} \frac{s_j(t) - s_i(t)}{\left|s_j(t) - s_i(t)\right|^3} dt^2 \tag{2}$$

with $p, q > 0$.

And, as apparent from the experimental results, the best results are for $p = q = 0$, which gives place to the unitary Markov model with

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} \frac{1}{m_i(t)} \frac{s_j(t) - s_i(t)}{\left|s_j(t) - s_i(t)\right|^3} dt^2 \tag{3}$$

3. New gravitational clustering algorithm

In this section we present the main content of our work, the new generalization of cluster gravitational algorithm which generalizes the one described above, in subsection 2.3. Instead of $\frac{1}{m_i(t)}$ in the unitary Markov model (3), we take $O\left(\frac{1}{m_i(t)}, 1\right)$, where $O: [0,1]^2 \to [0,1]$ is an arbitrary overlap function.

Since an overlap function is given on the unit square $[0,1]^2$, a normalization is needed. It can be realized by using e.g.

$$O\left(\frac{{m_i}^{p-1}}{n^{p-1}}, \frac{{m_j}^q}{n^q}\right)$$

With $p, q > 0$. However, we focus on the case p, q going to zero (the best referred case in original algorithm), it is hold that this function is always greater or equal than $O\left(\frac{1}{m_i(t)}, 1\right)$. So, let us have an overlap function $O: [0,1]^2 \to [0,1]$. When a configuration of n particles is given: $p_1, \ldots, p_n$, and their positions $s_1, \ldots, s_n \in \mathbb{R}^n$, the algorithm we design is:

1. Initially we assign a mass 1 to each particle p_i.
2. We fix real positive parameters ϵ and δ
    - We utilize δ for determining the actual time step of the actual dt. Indeed, in the lapsing time $[t, t + dt]$, the most rapid particle moves by δ.

- If in a moment two particles find themselves in a distance less than ϵ, we unify them in one with mass equal to sum of masses of both of them and position done by their center of gravity.

3. Initial time $t = 0$ is set.
4. We repeat following steps (i)-(iv) until a single particle remains.

(i). In each time interval $[t, t + dt]$, for each particle i we compute its movement influencing function:

$$g(i,t,dt) = \frac{1}{2} G \sum_{j \neq i} O\left(\frac{1}{m_i(t)}, 1\right) \frac{s_j(t) - s_i(t)}{\left|s_j(t) - s_i(t)\right|^3} dt^2 \tag{4}$$

Where G is a positive constant.

(ii). First we find the fastest particle p_I,

$$I = \arg\left(\max_i \{g_i(t)\}\right),$$

then we gain the length of current time step from the maximal shift restriction δ stipulated:

$$|g_I(t)| = \delta \Rightarrow dt(t) \text{ quantified}$$

and finally the new position of each particle i is:

$$s_i\big(t + dt(t)\big) = s_i(t) + g_i(t)$$

(iii). We raise t to $t + dt$.

(iv). If two particles i and j are in a distance less than ϵ, their unification is done as explained above.

Finally we have just one particle. The duration of the entire process is denoted by T. The evolution of the process is recorded in the sequence $\left\{t_{n-k_1}, t_{n-k_2}, \dots, t_{n-k_q} = t_1\right\}, 0 < k_1 < \cdots < k_q < n$; here we record all time moments when a change in the number of particles happened. The number of particles can decrease by more than one within one level of configuration. The current number of particles can be referred by indexing. Then

- In time interval $\left[t_n = 0, t_{n-k_1}\right]$ the number of particles is n.
- In time interval $\left[t_{n-k_1}, t_{n-k_2}\right]$ the number of particles is $n - k_1$.
- …
- In time interval $\left[t_{n-k_{q-1}}, t_{n-k_q} = t_1\right]$ just two particles remain.
- In $[t_1, \infty]$ all data are gathered in one cluster, one particle.

Then 100% of the total time $T = R_1 = t_1 - t_n$ can be split at q parts with q values of relative times assigned.

3.1. About particle movement governing mapping contraction

As the movement of particles system is driven by Newton gravitational force and the Markov process is considered, i.e. no history influences are included, in order to prove the convergence of the algorithm it is sufficient to prove the contractiveness of the system.

Physical interpretation gives us the imagination of the contraction of the system with proceeding time. Each single particle is driven towards the center of gravity of all the other particles. The restriction of maximal shift to sufficiently small δ prevents a possibility of escaping particles. We have to prove that convex envelope of the particles is shrinking.

We can rewrite (4) coordinate-wisely

$$g_i^d(t) = \sum_{j \neq i} O\left(\frac{1}{m_i(t)}, 1\right) \frac{s_j^d(t) - s_i^d(t)}{\left|\overrightarrow{s_j}(t) - \overrightarrow{s_i}(t)\right|} \frac{1}{\left|\overrightarrow{s_j}(t) - \overrightarrow{s_i}(t)\right|^2} dt^2 \tag{5}$$

As $\frac{1}{2}G$ has no influence on the contractiveness proof, it can be omitted in the following considerations from the formula (4).

As mentioned above, the fastest particle I moves by δ in one time interval. That determines the longitude of the specified $dt(t)$

$$\delta = |g_I(t)| = O\left(\frac{1}{m_I(t)}, 1\right) \sum_{j \neq I} \frac{1}{\left|\overrightarrow{s_j}(t) - \overrightarrow{s_I}(t)\right|^2} dt^2 \Rightarrow$$

$$dt^2(t) = \frac{\delta}{O\left(\frac{1}{m_I(t)}, 1\right) \sum_{j \neq I} \frac{1}{\left|\overrightarrow{s_j}(t) - \overrightarrow{s_I}(t)\right|^2}} (> 0) \tag{6}$$

which for e.g. $O(x, y) = xy$ yields

$$dt^2(t) = \frac{\delta m_I}{\sum_{j \neq I} \frac{1}{\left|\overrightarrow{s_j}(t) - \overrightarrow{s_I}(t)\right|^2}} \tag{7}$$

2 dimensional interpretation:

We deal with N particles, with position vectors in a time instant t: $\overrightarrow{s_i}(t) = \left(x_i(t), y_i(t)\right)$. First we create the convex hull of the space occupied by all particles and immerse the convex hull into a minimal covering rectangle $\langle mX(t), MX(t)\rangle \times \langle mY(t), MY(t)\rangle$, where

$$mX(t) = \min_i\{x_i(t)\}, mY(t) = \min_i\{y_i(t)\} \tag{8}$$

$$MX(t) = \max_i\{x_i(t)\}, MY(t) = \max_i\{y_i(t)\} \tag{9}$$

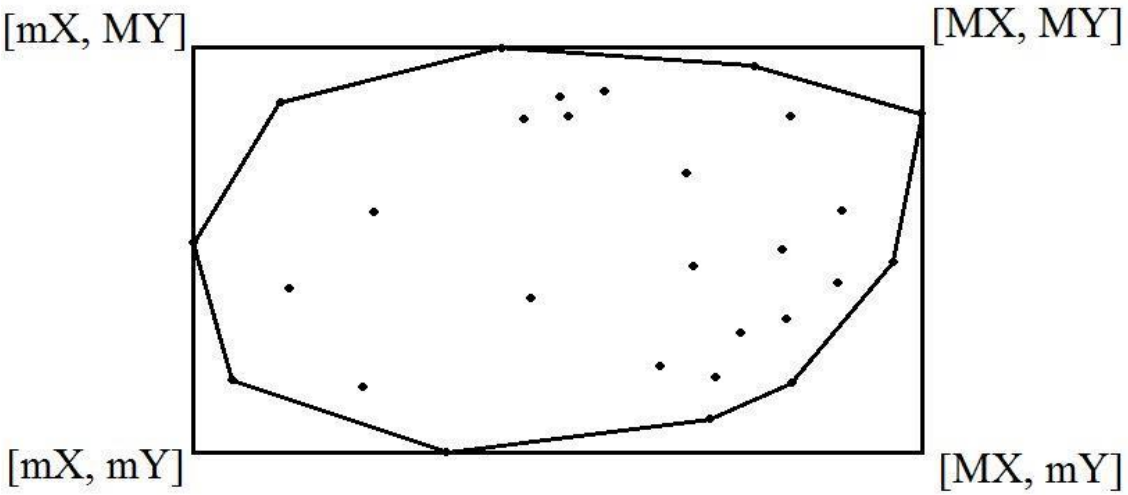


Figure 1 Convex hull of the particle set and minimal rectangle over the convex hull

As depicted in Fig. 1. Then the contraction of the rectangle will proceed together with contraction of the convex hull.

Theorem 3.1. *Supposing $O(x,y)$ is an overlap function given by Definition 2.5, the minimal covering rectangle shrinks with time lapsing when the particles motion is driven by (5).*

Proof. It is to be proven that

$$mX(t) < mX(t + dt(t)) \tag{10}$$

$$mY(t) < mY(t + dt(t)) \tag{11}$$

$$MX(t) > MX(t + dt(t)) \tag{12}$$

$$MY(t) > MY(t + dt(t)) \tag{13}$$

For $\vec{s}_{\tilde{m}} = (mX, mY)$ we have

$$g_{mX}(t) = O\left(\frac{1}{m_{mX}(t)}, 1\right) \sum_{j \neq mX} \frac{s_j(t) - s_{mX}(t)}{\left|\vec{s}_j(t) - \vec{s}_{\tilde{m}}(t)\right|} \frac{1}{\left|\vec{s}_j(t) - \vec{s}_{\tilde{m}}(t)\right|^2} dt^2 > 0$$

And

$$mX\big(t + dt(t)\big) = mX(t) + g_{mX}(t)$$

Implies (10).

Proof of (11)-(13) is analogous.

More dimensional interpretation:

It is apparent, that by proceeding coordinate by coordinate we can easily enhance two dimensional case to D dimensional, $D > 2$.

Remark.

- When proceeding in time, there are some intervals $[t, dt(t)]$ in which no topological change of system happens and some, in which some merges of particles are done. As

far as the convergence concerns, there is no difference between these two cases. Since the new position of the resulting particle originated from several ones being too close each to other (distance < ϵ) will be situated in the center of their gravity, none of their coordinates over-crosses the convex hull, likewise the minimal covering rectangle pertaining to this time interval.

- Author in [9], reasoning by the excellent experimental results, restricts the generalized Markovian model to the case $p \to 0, q \to 0$. It means that he diminishes the influence of masses of all attractors of i^{th} particle. As performed above, an overlap function generalizes this expression, convergence being ensured, as proven theoretically. However, if the resulting value of the overlap function is too small, it can bring slow convergence or rounding errors caused inexactness due to very small values of dt(t).
- As $dt(t)$ for each t is set in accordance with O, see (14) we do not utilize (O4). Neither do we utilize (O3) nor (O5). It's necessary to study other possible changes in base to the made experimentation.

$$g_{mX}(t) = \frac{\delta}{O\left(\frac{1}{m_I(t)}, 1\right)\sum_{j \neq I} \frac{1}{\left|\overrightarrow{s_j}(t) - \overrightarrow{s_I}(t)\right|^2}} O\left(\frac{1}{m_{mX}(t)}, 1\right) \sum_{j \neq mX} \frac{s_j(t) - s_i(t)}{\left|\vec{s}_j(t) - \vec{s}_i(t)\right|} \quad (14)$$

4. Experimentation

4.1. Used overlaps

The overlaps that have been used with success are the following:

$$1. \quad O(x, y) = (x * y)^p \to O\left(\frac{1}{m_i}, 1\right) = \frac{1}{{m_i}^p}$$

$$2. \quad O(x, y) = \frac{\sqrt{x}}{\sqrt{x} + \max(1-x, 1-y)} \to O\left(\frac{1}{m_i}, 1\right) = \frac{\sqrt{\frac{1}{m_i}}}{\sqrt{\frac{1}{m_i}} + \max\left(1 - \frac{1}{m_i}, 0\right)}$$

Other overlaps were tested, but these were the ones that offered the most interesting results.

4.2. Algorithm's behavior

In order to visually represent the behavior of the algorithm, a small dataset consisting on 9 examples will be used. Firstly, the classic algorithm, the unit Markovian model, will be tested. Later, the modified version will be applied, using both overlaps 1 and 2, and the results will be analyzed. The used dataset is represented in Fig. 2.

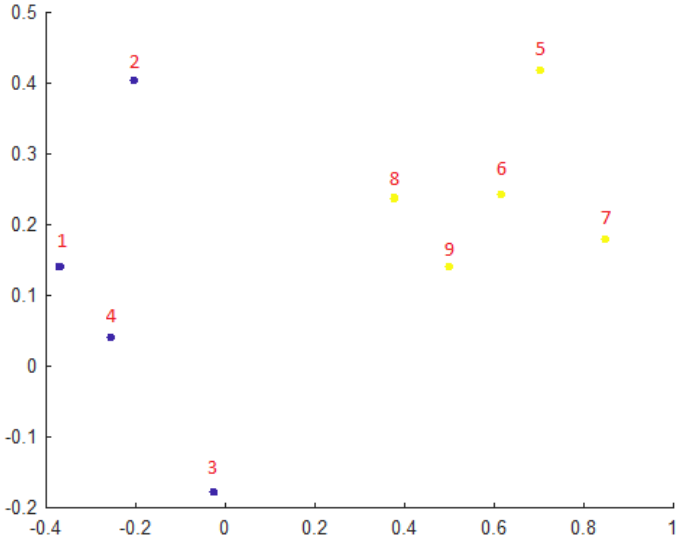


Figure 2 Visual representation of the dataset "some points"

Fig. 3 represents the movement of the particles during the execution of the algorithm. Each dot shows the position of a particle after a given iteration, being the ones surrounded by a red ring the original ones. Collisions are represented by a red mark, and come with the number of the iteration in which they have taken place. For these tests, values of $\delta = 0.1, \epsilon = 0.2$, which turn out to be good enough, were set.

It can be seen that, before entering in the algorithm's loop, 3 fusions have already taken place. This happens because the pairs of particles (1, 4), (5, 6) and (8, 9) start within distance ϵ, and thus get fused. It's important noticing how the resulting particles move around half of the distance than the rest, which occurs because, as seen in equation (3), the model punishes the most massive particles. After iteration 3, moment in which both main clusters are created, it takes 4 more iterations for them to collide, during which the mass 4 particle is the one that moves the most.

The particles' mass also determines the collision's position, that takes place in their center of mass. It can be seen when comparing the position of the resulting particles after the collisions of iteration 0, equidistant to each of them, and the rest, that are created closer to the most massive particle.

The elapsed time between iterations is also an important parameter, since it's the one that determines the optimal number of clusters. This is represented in Fig. 4, which shows that it takes around 80% of the total time for the two main clusters to join. It's also interesting to see that the second longest lapse of time takes place when there are 4 clusters, composed by

particles (5, 6, 7, 8, 9), (1, 4), (2) and (3), which makes sense since particles 2 and 3 are fairly far from the rest of their cluster, and could form their own.

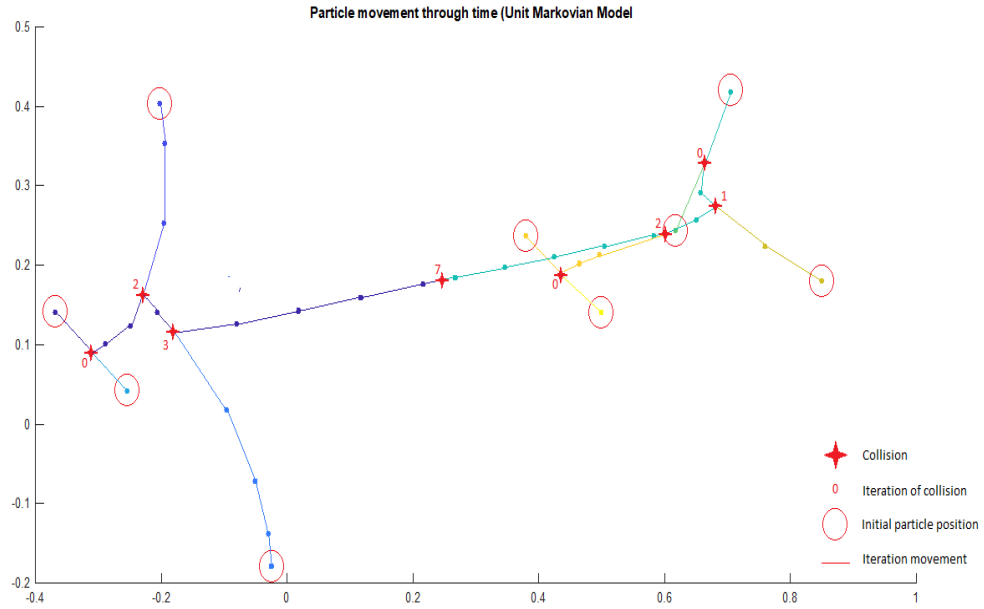


Figure 3 Evolution of the algorithm (Unit Markovian model)

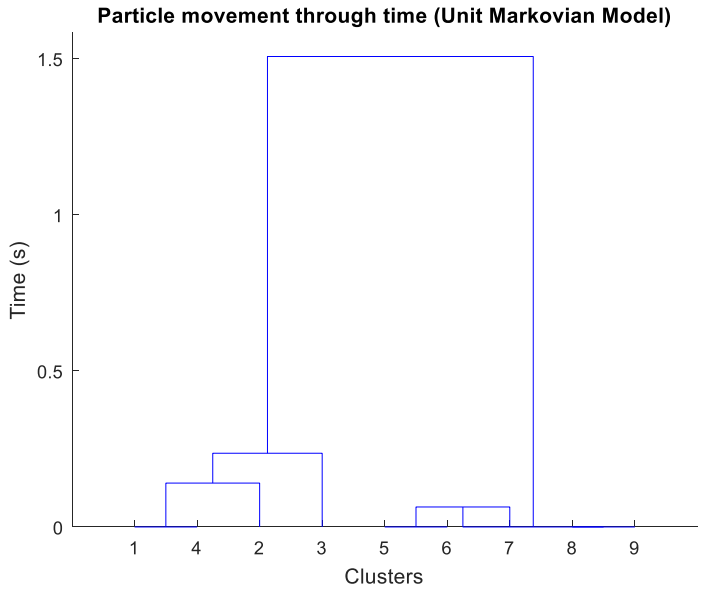


Figure 4 Elapsed time between collisions (Unit Markovian model)

If we start analyzing the behavior of the algorithm when using overlap 1, it can be seen in the Fig. 5 that, as the parameter p increases, so does the time between collisions. This increase is exponential however, and has its root in the fact that p gets used as exponent for the particle's mass in equation (4). If there was already some punishment to the most massive particles when using the unit Markovian model, it gets seriously aggravated when using this overlap. In fact, when p is big enough, the distance moved by those particles become negligible and they appear to stand still.

This can also be appreciated in the Fig. 6. Specifically, when $p = 20$ the colliding particles stay practically static until all remaining particles with less mass become fused. After iteration 5, when the cluster composed by (5, 6, 7, 8, 9) has been formed, it stands still until iteration 12, when the other cluster with less mass reaches it. It could be of interest noticing how in this type of situations the distance moved by a collision can be substantially bigger than the normal movement one.

All this can pose a problem when determining the optimum number of clusters, since the algorithm tends to spend the longest periods of time when two particles remain. When using this overlap, it would be recommended to avoid using this method, as it will most of the time return 2.

Overlap 2, however, works very similarly to the original algorithm, as it doesn't apply that much a punishment in the mass. Its behavior is shown in Fig. 7.

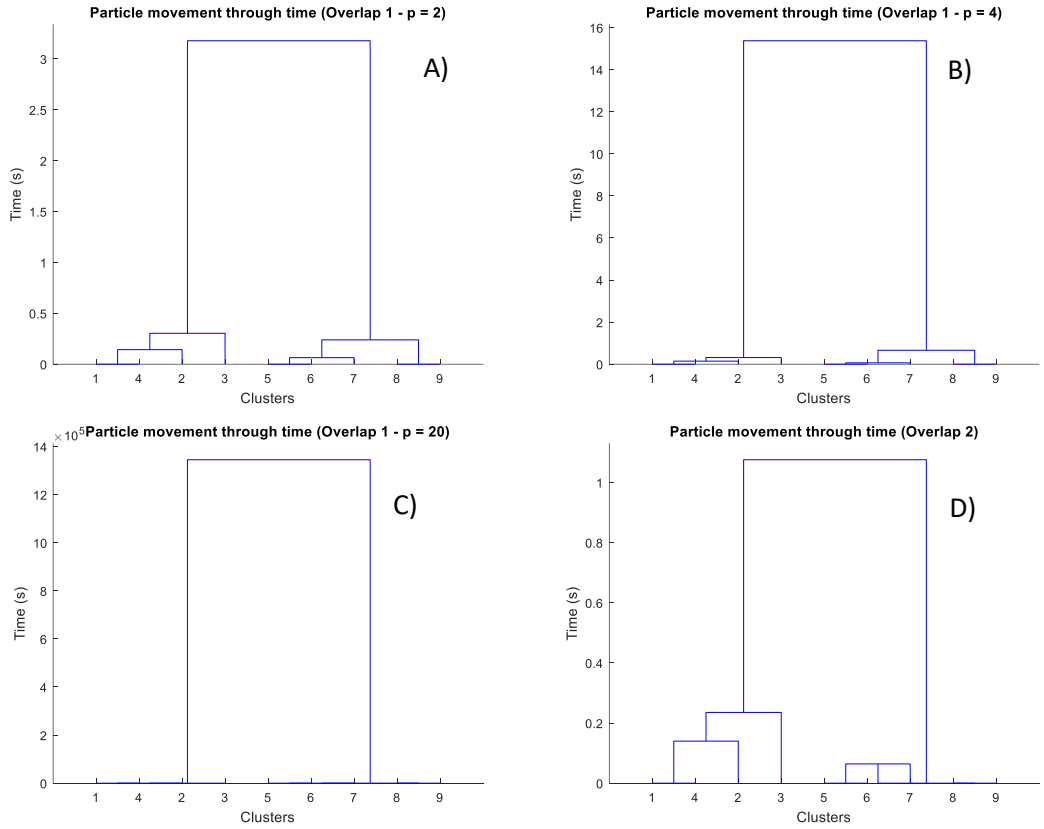
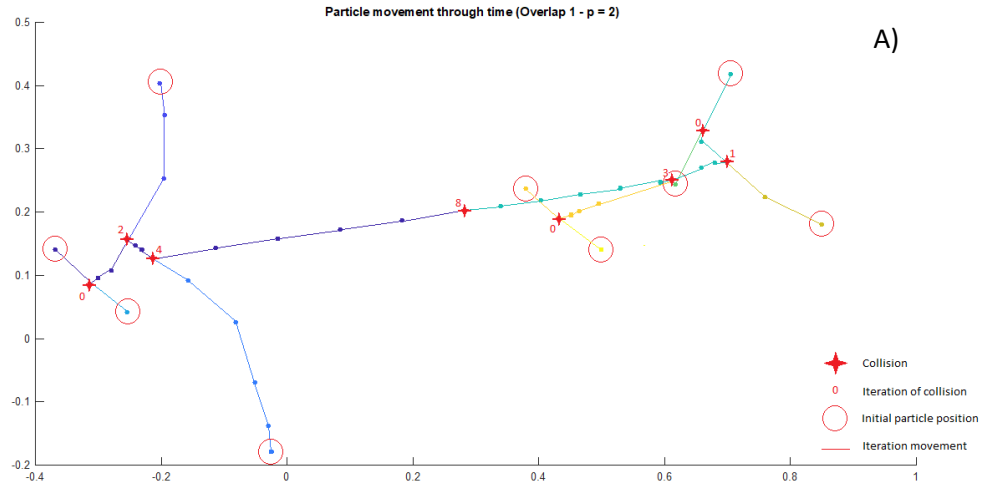
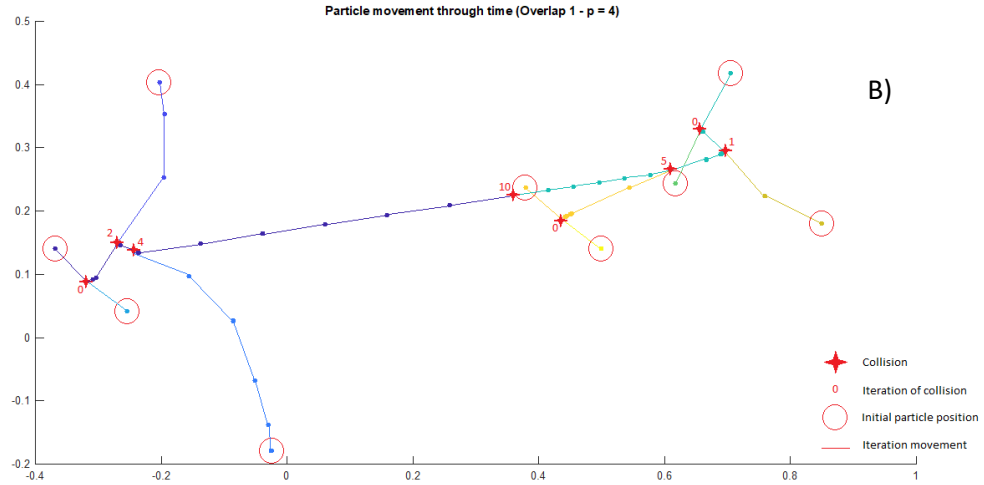


Figure 5 Elapsed time between collisions (A) Overlap 1, p = 2, B) Overlap 1, p = 4, C) Overlap 1, p = 20, D) Overlap 2)





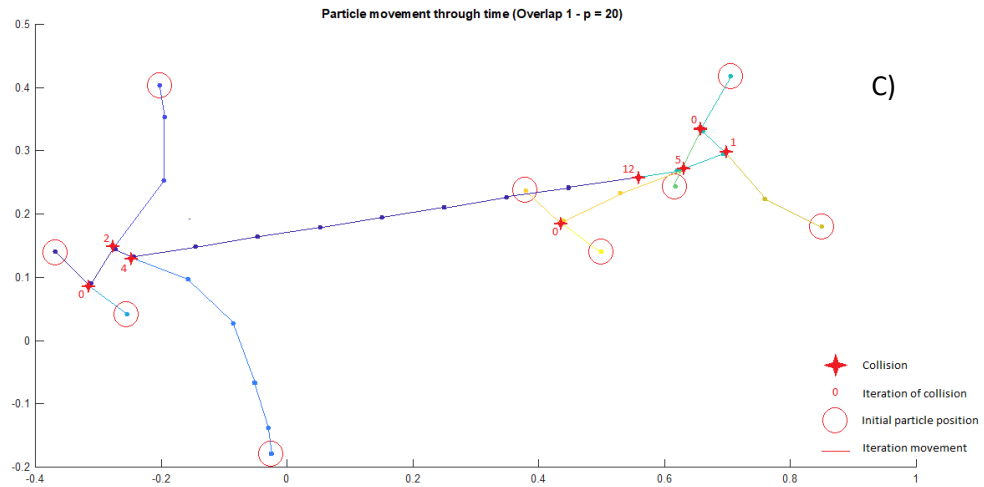


Figure 6 Evolution of the algorithm (Overlap 1 A) p = 2, B) p = 4, C) p = 20)

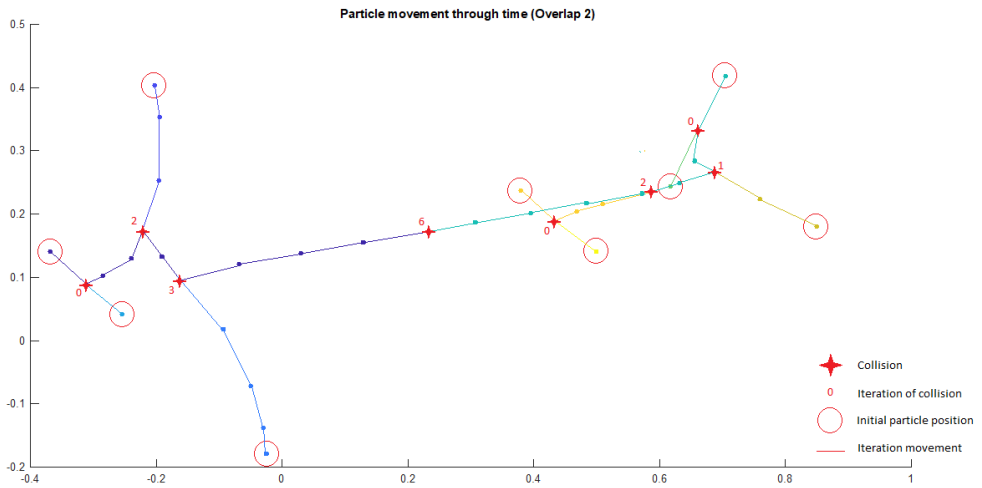


Figure 7 Evolution of the algorithm (Overlap 2)

4.3 Tests with synthetic datasets

The previous overlaps were tested, in conjunction with the classical algorithm, the K-means algorithm and the FCM (Fuzzy c-means) algorithm over a group of 2-D artificial datasets with different topologies, in order to find out their strengths and weaknesses, and with the objective to find the conditions in which best values are obtained.

Since one of the problems of the algorithm consists in determining the optimal values for the parameters δ and ϵ, both internal and external evaluation measures were used, to find if there's a correlation between both. If that were the case, then it would be possible to evaluate the resulting clusters taking only into account internal measurements in the situations where a priori classifications are unknown (which are the most common ones).

The used datasets are shown below:

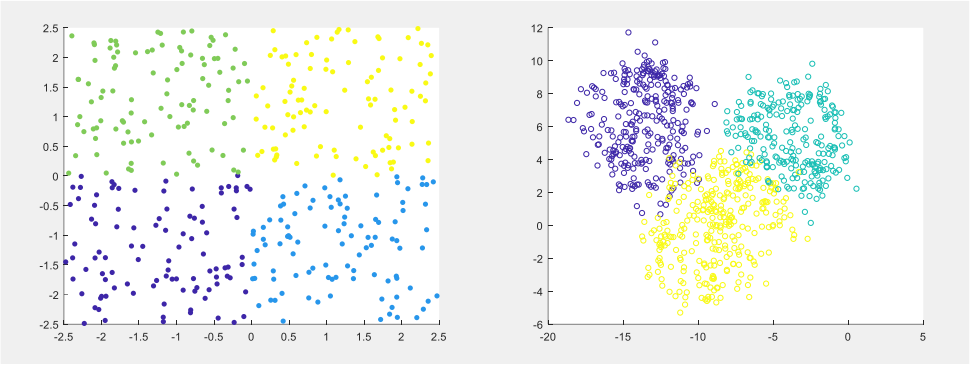


Figure 8 "4 squares" dataset

Figure 9 "3 close circles" dataset

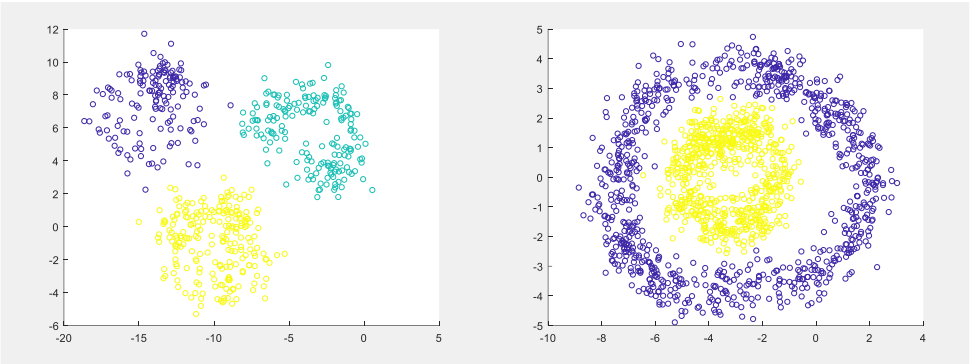


Figure 10 "3 circles" dataset

Figure 11 "2 rings" dataset

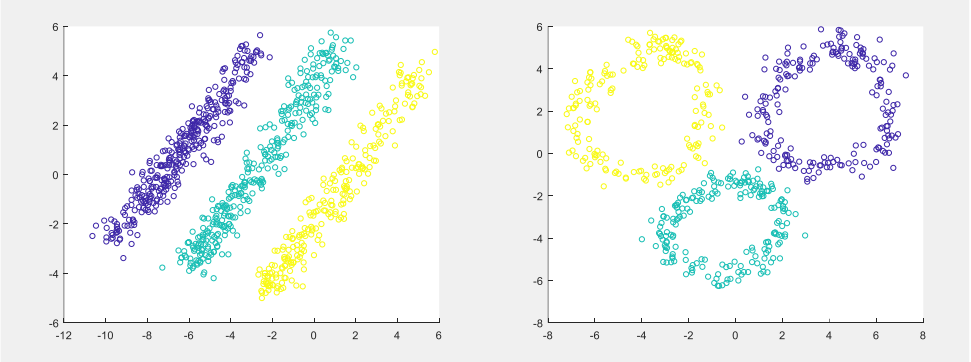


Figure 12 "3 stripes" dataset

Figure 13 "3 rings" dataset

The following tables synthesyze the obtained values, both maximum and on average, for the given datasets.

The difference between the winners of the first table (internal evaluation) and the rest (external evaluation) makes clear that there is no clear relation between both. That supposes a problem, since it makes it harder to find the best values for p, δ and ϵ. Nonetheless, the values obtained for the internal evaluation are mostly fine.

It can also be seen that overlap 1 can get great values, but is also the one which incurs in more variance. Overlap 2, however, obtains values similar to those of the unit Markovian model in average, while being able to improve them in many cases. However, it fails to get good values for those datasets in which overlap 1 gets the best values, which are commonly the ones with the most peculiar topology.

	OVERLAP 1 ($p = 20$)		OVERLAP 2		Unit Markovian Model		K-means	FCM
Dataset	Mean	Max	Mean	Max	Mean	Max	Max	Max
4_squares	0,508	0,682	0,577	0,682	0,578	0,682	**0,743**	0,736
3_close_circles	0,633	0,690	0,621	**0,691**	0,641	**0,691**	0,688	0,685
3_circles	0,848	0,861	0,859	0,860	0,858	0,860	**0,932**	**0,932**
2_rings	0,386	0,464	0,468	**0,497**	0,464	**0,497**	0,464	0,438
3_stripes	0,378	0,579	0,442	**0,596**	0,432	0,586	0,467	0,485
3_separated_rings	0,519	0,831	0,631	0,758	0,621	0,715	**0,874**	0,848

Table 1 Comparative of Dunn-Index among algorithms

	OVERLAP 1 ($p = 20$)		OVERLAP 2		Unit Markovian Model		K-means	FCM
Dataset	Mean	Max	Mean	Max	Mean	Max	Max	Max
4_squares	0,784	0,893	0,807	0,910	0,810	0,900	**0,988**	0,983
3_close_circles	0,859	0,946	0,856	**0,948**	0,852	0,920	0,901	0,901
3_circles	0,991	**0,998**	0,996	**0,998**	0,996	**0,998**	**0,998**	**0,998**
2_rings	0,577	0,628	0,622	**0,713**	0,624	0,705	0,570	0,570
3_stripes	0,670	**0,928**	0,564	0,642	0,569	0,690	0,697	0,690
3_separated_rings	0,806	**0,998**	0,788	0,916	0,784	0,916	0,960	0,960

Table 2 Comparative of purity among algorithms

	OVERLAP 1 ($p = 20$)		OVERLAP 2		Unit Markovian Model		K-means	FCM
Dataset	Mean	Max	Mean	Max	Mean	Max	Max	Max
4_squares	0,822	0,901	0,834	0,917	0,838	0,907	**0,988**	0,983
3_close_circles	0,843	**0,931**	0,840	**0,931**	0,836	0,902	0,884	0,884
3_circles	0,989	**0,998**	0,995	**0,998**	0,994	**0,998**	**0,998**	**0,998**
2_rings	0,507	0,532	0,533	**0,591**	0,533	0,584	0,501	0,501
3_stripes	0,718	**0,905**	0,626	0,702	0,633	0,719	0,762	0,754
3_separated_rings	0,812	**0,998**	0,741	0,900	0,735	0,900	0,949	0,949

Table 3 Comparative of Rand-Index among algorithms

5. Conclusions and future works

It has been proved that substituting the mass product in equation (3) by an overlap makes for a valid generalization method, which keeps ensuring the convergence of the algorithm. We've also stablished two overlaps which return interesting results and studied their strengths and weaknesses, which should be taken into account when deciding which of them apply to an specific dataset.

In particular, we've seen how, when using overlap 1 in conjunction with big values of p, great results are obtained in situations where all alternatives fail, but a lot of variance appears also. We've also observed how overlap 2 works very similarly to the original algorithm, but tends to improve the results obtained by it.

Overlaps have turned to be of interest in this case, but there's no reason to think that there couldn't exist other functions which could be used for substituting them. Thus, further study could be made in this sense, and other type of functions could be analyzed, in order to find other potential generalizations for the problem.

A matter we haven't gotten deep into is in that of the distance metric used in the algorithm, which employs the typical Euclidean distance metric. However, when working with problems of clustering, this metric is the one that typically determines the type of datasets that the algorithm will work best with. In this regard, further investigation will be also made, in search of a generalization that allows the algorithm to work properly when facing different topologies.

References

[1] H. Bustince, J. Fernandez, R. Mesiar, J. Montero, R. Orduna, *Overlap Functions*, Nonlinear Analysis, 72, 2010, pp. 1488-1499

[2] D. Gómez, J. T. Rodríguez, J. Montero, H. Bustince, E. Barrenechea, *n-Dimensional overlap functions*, Fuzzy Sets and Systems 287, 2016, pp. 57-75

[3] A. K. Jain, *Data clustering: 50 years beyond k-means*, Pattern Recognition Letters 31 (8) (2010) 651-666

[4] A. Jurio, H. Bustince, M. Pagola, A. Pradera, R. R. Yager, *Some properties of overlap and grouping functions and their application to image thresholding*, Fuzzy Sets and Systems 229, 2013, pp. 69-90.

[5] M. Minrov, D. Paternain, A. Jurio, J. Ruiz-Aranguren, Z. tak, H. Bustince, *Modifying the gravitational search algorithm: A functional study*, Information Sciences, 430, 2018, pp. 87-103.

[6] D. Paternain, H. Bustince, M. Pagola, P. Sussner, A. Kolesrov, R. Mesiar, *Capacities and overlap indexes with an application in fuzzy rule-bassed classification systems*, Fuzzy Sets and Systems 305, 2016, pp.70-94.

[7] E. Rashedi, H. Neamabadi-Pour, S. Sariazdi, *GSA: a gravitational search algorithm*, Information Sciences 179 (13), 2009, 2232-2248.

[8] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. Prakash Patel, A. Tiwari, E. M. Joo, D. Weiping, L. Chin-Teng, *A review of clustering techniques and developments*, Neurocomputing, 267, 2017, pp. 664-681.

[9] W. E. Wright, *Gravitational Clustering*, Pattern Recognition, Pergamon Press 9, 1977, pp. 151-166